Modeling Energy Efficiency in Distributed Home Environments

Helmut Hlavacs[1], Karin A. Hummel[1], Roman Weidlich[1],
Amine M. Houyou[2], Hermann de Meer[2]

[1]Institute of Distributed and Multimedia Systems, University of Vienna

[2]Department of Informatics and Mathematics, University of Passau

Technical Report, Number MIP-0713
Department of Informatics and Mathematics
University of Passau, Germany
December 2007

Modeling Energy Efficiency in Distributed Home Environments

Helmut Hlavacs[1], Karin A. Hummel[1], Roman Weidlich[1], Amine M. Houyou[2], Hermann de Meer[2]

[1]Institute of Distributed and Multimedia Systems, University of Vienna,
[2]Faculty of Computer Science and Mathematics, University of Passau

Abstract

In this paper we present a distributed approach for saving energy by sharing computing load in home networks. In our approach possibly thousands of home computers may cooperate and send each other tasks, which include services and applications running on a 24/7 basis. By concentrating as many tasks as possible on a small number of computers, idle computers may go asleep and thus consume almost no energy. We present the general architecture of our approach and analyze several 24/7 applications by modeling the potential energy consumption with and without application sharing.

1 Introduction and Motivation

Modern home environments are envisioned as multimedia homes consisting of a multitude of networked devices, intelligent home appliances, and sensors. In combination with the home PC, these devices form the *home network*, which is then connected to the Internet. Furthermore, more and more *always-on* services are requested by home users. Always-on services include multimedia services and home management with/without remote control, in addition to distributed online applications (such as online gaming or P2P file-sharing). These current and future always-on services rely on home computers running on a 24/7 basis. Nevertheless, many of these services only require a limited portion of the available resources in modern PCs, and thus result in massive energy wastage world wide. For example, a low-cost PC consumes about 100 W if switched on, a multimedia PC consumes 148 W, compared to a few watt if the computer is hibernating.[1]

The role of energy saving in the IT sector has been attracting a lot of efforts, both with respect to environmental concerns and increasing cost of energy. In addition to increasingly growing CO_2 production, in server farms long term electricity consumption costs are steadily approaching the cost of the hardware itself [12]. A similar trend is seen in the home. However, to the best of our knowledge, most existing efforts to reduce energy consumption in IT industry have focused on localized solutions (e.g. making hardware energy efficient or running a hardware at high utilization due to virtualization).

To extend energy efficiency concepts to the home environment, several challenges need to be addressed. Among these are reducing the number of active computing resources.

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

Whereas P2P resource sharing offers some insights in how to mediate resources between homes, it does not provide the full fitting answer here. P2P related research has not addressed energy-aware systems before. Meanwhile, an energy-aware mediation requires a complex management system to tightly track and control large distributed resources. Modeling home services and applications – as is done in this paper – in terms of their energy consumption comprises a first step to build such a system.

The different existing techniques relating to energy efficiency are looked at in Section 2. In Section 3 the formal definition of distributed energy efficiency is given. The role of P2P for energy-aware home resource sharing is explained in Section 4. Qualitative and quantitative models showing how energy can be saved by our approach follow in the remaining sections. An analytic model of remote video encoding is given in Section 5. Then follows a Markovian analysis of a download sharing scenario (based on a file-sharing system) in Section 6. In Section 7 the feasibility of hosting avatars is demonstrated. A remote home management systems relying on sensor networks is modeled in Section 8. Finally, Section 9 concludes the paper.

2 Related Work

The always-on home PC with a DSL high data rate connection to the Internet imposes a new dimension of cost, namely energy consumption. Koomey [19] reports a doubling of energy consumption from 2000 to 2005 of volume, mid-range, and high-end servers in the U.S. and worldwide. A similar tendency could be expected for always-on home PCs. End devices in the home are contributing to a large portion of the electricity consumption growth in the EU for instance [2]. Computer power can be saved by means of various well known techniques. First, the processor can be powered down by mechanisms like SpeedStep [14], PowerNow, Cool'nQuiet, and Demand Based Switching. These measures enable slowing down the clock speeds (Clock Gating), or powering off parts of the chips (Power Gating), if they are idle [13, 22]. By sensing user-machine interaction, different redundant hardware parts can incrementally be turned off or put on hibernating mode (display, disk, etc.). These techniques are usually applied to notebooks and mobile devices but can be used for desktop PCs as well. Further energy saving methods relate to communication energy cost. Battery lifetime of wireless devices can be improved by addressing several layers of the network protocol stack (Physical, Data Link (LLC, MAC), Network, and Transport layers), operating systems, middlewares, and applications [15, 24]. All of the above techniques must be regarded as *local* energy saving techniques. Further green computing recommendations are also incited by the US Energy Star[2] and the European TCO[3] Certification, who rate IT products (mostly monitors) for their environmental properties. Energy has also been addressed in some projects, like the EU funded project *SmartHomes* [7], coming up with a simulation approach for large-scale energy management, and the approach of remote monitoring of e-energy services in [16].

In our scenario however, the home PC is not a stand-alone machine. In [11], a distributed P2P approach to interconnect home networks is proposed similar to a PlanetLab [20]. Therefore, a global approach was taken to save energy in a large distributed architecture. Similar energy saving measures in distributed systems are found in data centers. There, servers [12] are consolidated, via a central management process, to increase their computational load [18]. Therefore the work load per consumed energy is increased, which is the measure used for energy efficiency [21]. A further analysis of the home environment, however, requires looking at real home applications, and taking their

[2]http://www.energystar.gov/
[3]http://www.tcodevelopment.com/

requirements into account. Then, energy consumption models based on the architecture, proposed in [11], can be better evaluated.

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) in order to reduce energy consumption. For reasons of robustness and scaling, a distributed solution is proposed considering each of these characteristics in a P2P network. 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.

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 should be minimized. The basic formula 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 [21] 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 $E(T) \neq 0\,\text{kWh}$. If the energy consumption can be reduced by sharing, then the energy efficiency will increase.

Additionally, the system assures a certain degree of trust in the non-functional characteristics of home services. 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 (Fairness means that each home should consume approximately as much as it contributes to the system).

Based on these basic energy formulae, in this paper 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.

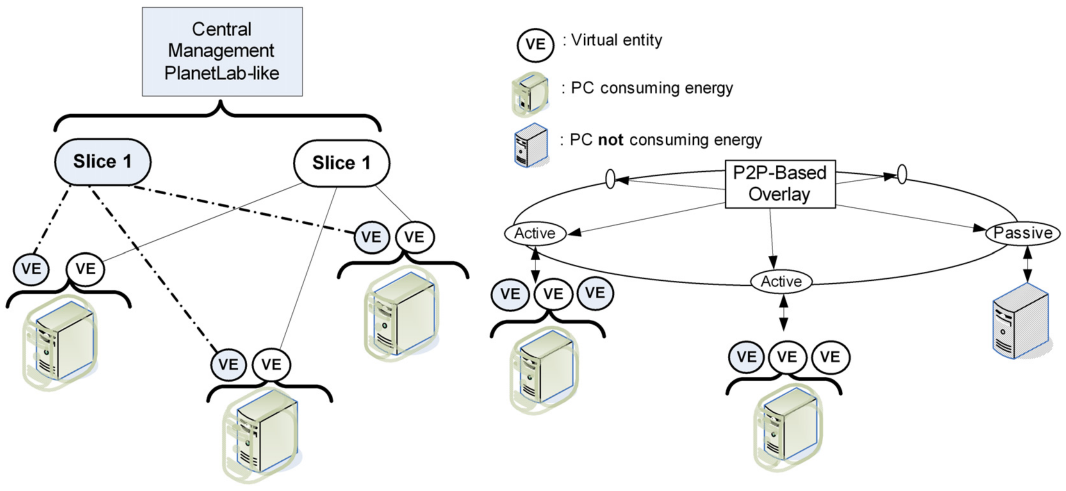

Figure 1: a) PlanetLab architecture and b) Energy-aware P2P management.

4 Virtual Home Environments

The architecture which this paper is based on already has been presented in [11]. In this section the main elements are summarized to help understand the running environments of the modeled applications.

It is worth noting the similarity to the PlanetLab approach of virtualization (See Fig. 1a). However, PlanetLab has not been developed taking energy efficiency into account, neither is the management approach scalable for the type of applications predicted for the home environment. A PlanetLab type of architecture (as shown in Fig. 1a) can be summarized as a platform connecting any type of computers (i.e., from home PCs to large server machines) via the public Internet and in a P2P manner. Each computer is split into smaller virtual entities (e.g., virtual machines). Although the participating nodes share their resources equally, the management and trust are centralized in PlanetLab. Especially when adding hardware resources to the system and allocating virtual resources to users, a central administrator is involved. The central authority provides a user with a slice, which is a collection of virtual entities (one on each PlanetLab node). The user must centrally upload his/her code (via SSH for instance) by addressing each node individually. It is, therefore, up to the user to install the code on all machines or not, creating a great deal of redundancy, and thus rendering the system itself energy inefficient.

Differently to PlanetLab, our architecture is based on a distributed management approach (as shown in Fig. 1b). The connected homes form a distributed pool of virtual resources similar to PlanetLab, but the architecture uses a distributed management system to allocate resources to home PCs in a dynamic and flexible manner. The main goal of the architecture is to achieve a concentration of load and communication in parts of the inter-connected homes (seen as a distributed infrastructure), while idle resources can be hibernated. In this way, the total consumed energy is reduced.

The intelligence of the distributed management is situated in each contributing node in a P2P manner. There is no central control entity involved in the decision process. The control mechanism is distributed between peering homes. To allow energy saving, a distinction is made between active and passive peers (contributing homes, shown in

Fig. 1b). The *active peers* are homes which contain a home PC that is on (i.e., with full energy consumption). The *passive peers* are those homes where only the gateways are running. A *gateway* could be seen as a Linux-based diskless computer with small energy needs. The gateway can maintain a permanent entry in the system wide overlay and represent its particular home.

The P2P-based overlay provides services like identifying other peers and controlling the power consumption cycles per peer. It provides a system wide distributed database for storing node and user statistics persistently. It also manages resource requests per application or service, which is then assigned to a subset of the online computers. Above it, optimization models constitute the decision basis of the system. They can be roughly divided into the following submodels. First, the *energy efficiency* submodel which, upon a request for resources, aims at minimizing the global energy consumption while maximizing global energy efficiency, as defined in Section 3. Second, the *fairness* model uses statistics on how much each home contributed to the system for a fair share of energy saving. The statistics can be managed using a distributed hash table (DHT). The *availability* submodel decides on how to replicate the service depending on the application needs and failure rates. The *privacy* model tries to maximize the degree of privacy for a service by providing chains of anonymity mixers, for instance.[4] As for the *quality of service* (QoS) model, home services can only be hosted on parts of the network that offer the right QoS conditions. Some other functionalities of the management system have to take QoS metrics into account, like when moving a virtual machine or for computing the costs for privacy (i.e., delay and throughput of a shorter or a longer chain of anonymity mixers). Finally, the *security* model is based on P2P-based key exchange and encryption protocols. Voting and reputation models will be the main elements of the distributed trust chain. Security is no longer based on a hierarchical centralized chain of trust as in PlanetLab, but rather a distributed one with local access control rules, for instance. In the following sections we now present the potential energy saving for a set of applications.

5 Remote Video Encoding

A scenario where raw CPU power is shared is given by remote video encoding. Think of a usual video production cycle as observed by many owners of camcorders. Users first create the footage, which is then edited, for example by using programs like Pinnacle's *Studio*.[5] Users then create a new video film by re-encoding the edited footage, for example into a DVD or an XVID film. This video encoding takes a lot of computing power and time.

In the presented scenario *remote video encoding*, owners of weak PCs can create edit points and then send the videos to much stronger PCs in other homes, where the videos can be encoded much faster. A necessary prerequisite is that the camcorder already produces compressed footage of reasonable size. In contrast to DV-AVI as produced by older DV camcorders, which yields as much as 14 GB per hour, modern camcorders save their videos on flash cards, hard disks or directly on DVDs, already compressed in either MPEG-2 or MPEG-4.

Assume a video of size L_1 MB which is produced in home A. User A then uses his/her editing software to create cut points and saves it together with the footage. The resulting new video then is assumed to be of size L_2 MB. Let R be the (symmetric) end-to-end bandwidth between home A and home B in MByte/s. Let T_A be the time it takes to encode the video in home A, and T_B be the time it takes to encode the video in home

[4] http://freenetproject.org/

[5] http://www.pinnaclesys.com/

B. Furthermore, let W_A and W_B be the power consumption rates of the PCs in the respective homes (in watt). Then the total energy consumed by home A when encoding the video is $E_A = T_A W_A$ joule, while the total power consumed when encoding the video in home B is

$$E_B = \frac{L_1}{R}(W_A + W_B) + T_B W_B + \frac{L_2}{R}(W_A + W_B) \text{ joule.}$$

For given L_1, W_A, W_B, T_A, T_B and R, and estimated L_2, the energy is saved if $E_B < E_A$, which leads to

$$\frac{RT_B + L_1 + L_2}{RT_A - L_1 - L_2} W_B < W_A.$$

Setting $T_B = \alpha T_A$ and $W_B = \beta W_A$ and solving for R yields

$$\frac{\beta + 1}{1 - \alpha\beta} \frac{L_1 + L_2}{T_A} < R. \tag{2}$$

Equ. (2) states the network bandwidth necessary to save energy. For $\alpha\beta = 1$ there is a singularity, meaning that energy can only be saved if $\alpha\beta < 1$. The nearer $\alpha\beta$ approaches 1, the higher the bandwidth must be in order to save energy. [10] shows some performance gains for different CPUs when encoding video, proving that α can be even lower than 0.5, thus justifying our choice for $1/2 \leq \alpha < 1$. Fig. 2 shows the required minimum bandwidth (Mbit/s) for $T_A = 1.5\,\text{h}$, $L_1 + L_2 = 1.5\,\text{GB}$, and $\beta = 0.8, 1$, and 1.2. It can be seen that the minimum bandwidth quickly rises near $\alpha\beta = 1$. Although in this example,

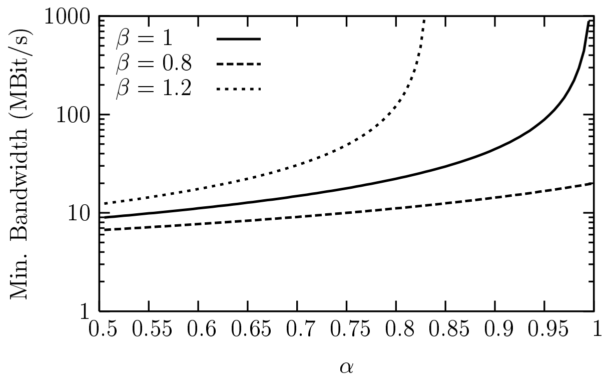

Figure 2: Minimum bandwidth required for saving energy.

10 Mbit/s seems to be a quite high demand, as a matter of fact, the current *average* access bandwidth as for example installed in Japan in 2007 [1] already is 60 Mbit/s. Furthermore, due to dark fiber and fiber to the home, in the near future we will find 100 Mbit/s and more in many access networks throughout the western countries. What is much more important is a good performance/power ratio for future CPU generations for pushing $\alpha\beta$ as low as possible.

The above described scenario can be improved significantly by restricting the sending of the videos to times when at least one of the computers is actually used by its owner. A computer running, for example, Office or a Web browser can easily download data in the background without decreasing the application's performance. This is not true if the computer is concurrently encoding a video, which would negatively affect the user's experience. This way, either L_1 or L_2 can be set to zero in (2), thus effectively halving the necessary bandwidth R.

6 Download Sharing

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 modeled 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. Additionally, more and more ISPs apply shapers to P2P traffic and thus artificially limit B_l for their customers.

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 $B_d = 4\,\text{Mbit/s}$, and each download on average consumes $B_l = 200\,\text{Kbit/s}$, 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 $F = 100\,\text{MB}$, and $B_l = 200\,\text{Kbit/s}$, 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 more complicated M/M/M queue would be necessary. The process states and transition rates are shown in Fig. 3.

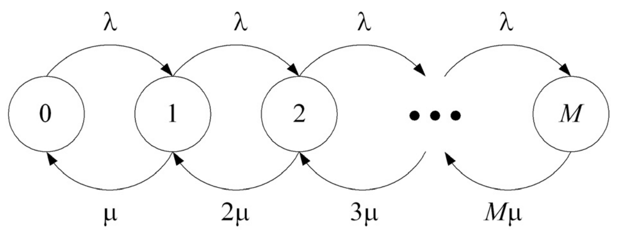

Figure 3: Birth-death process for local downloads on one single computer.

Simple analysis shows that the probability π_k for being in state k is given by [4]

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

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

$$N_{local} = N \left(1 - \frac{1}{1 + \sum\limits_{k=1}^{M} \frac{1}{k!} \left(\frac{\lambda}{\mu} \right)^k} \right). \tag{3}$$

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. 4.

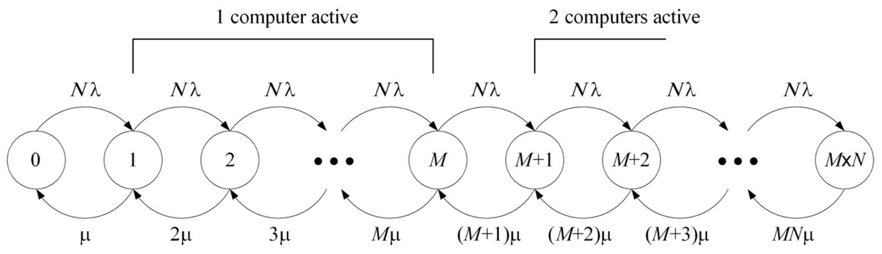

Figure 4: Birth-death process for simultaneous downloads of N computers.

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

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

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 $\pi_k, 1 \leq k \leq M$, and in general the probability for needing exactly a active computers is therefore the sum of the $\pi_k, (a-1)M + 1 \leq k \leq aM$. For computing the expectation N_{ideal} of a, we derive

$$N_{ideal} = \sum_{a=1}^{N} a \sum_{k=(a-1)M+1}^{a\,M} \pi_k. \tag{4}$$

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

$$N_{corr} = N_{ideal} + \frac{B_l}{B_u} \sum_{k=1}^{MN} k\, \pi_k \, \frac{N - \lceil k/M \rceil}{N}. \tag{5}$$

Equ. (5) 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.

If additional resilience is necessary then it is even thinkable to download each request two times, thus doubling the necessary energy requirements.

Fig. 5 shows results for $N = 1000, F = 100\,\text{MB}$, $B_d = 4\,\text{Mbit/s}$, $B_l = 200\,\text{Kbit/s}$, 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 (3) and the corrected (Corr) case (5). Even when downloading every file twice to increase error resiliency (Corr (2)), almost 50% of the energy can still be saved. 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 without error resilience, and 11.4 kW with error resilience. 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 (5), 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. 6. The energy efficiency of the sharing scenario (Corr) is clearly much better than the one for the scenario without cooperation (Local), even when enforcing error resilience (Corr (2)). 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.

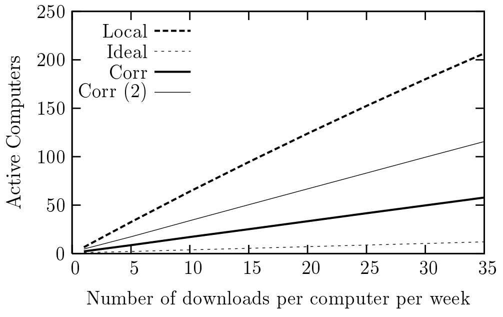

Figure 5: Number of active computers.

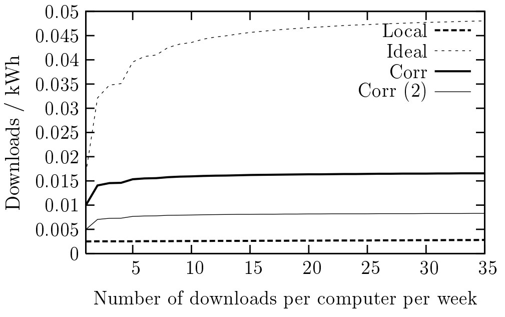

Figure 6: Energy efficiency η.

It must be noted that the corrected model does not take into account several details, such as representative file size distribution, protocol overhead and upload queuing. The latter must be accounted for if $B_d > B_u$, since here the upload queue of a single node may grow too large. 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.

7 Avatar Hosting

Recently Massive Multiplayer Online-Roleplaying Games (MMORPGs) became very popular.[6] Examples for MMORPGs include *World of Warcraft*, *Everquest 2*, *EVE Online*, *Second Life*, *Dungeons & Dragons Online*, *The Lord of the Rings Online*, etc. Similarly to First Person Shooters (FPSs) like *Quake 3*, *Counter Strike*, *Halo* etc., the player may put a considerably amount of time in game practice, resulting in long running PCs and thus high energy consumption. In MMORPGs the gaining of experience to obtain skills and collecting items causes additional uptimes.

[6]http://www.mmorpg-planet.de/

In FPSs artificial players are called *bots*. Similar to a Non Player Character (NPC) in classical role-playing computer games, bots were introduced to act as enemy if not enough human opponents are available. This also enables to play a multiplayer FPS offline against computer enemies only.

A big difference between humans and bots is the rate of commands per time unit in large-scale [5]. Firstly a bot perceives the environment by inspecting server packets. This permits the bot to react immediately to changes in the game world, whereas a human needs to recognize the changed game situation on the screen and then use an input device to release a command to the server. Secondly periodicity of server traffic propagates directly into the traffic pattern produced by bots. The burstiness of bot traffic is lower than the burstiness of heavy-tailed traffic as caused by human activities. Thirdly a bot is less sensitive to network conditions in contrast to humans. The rate a bot releases commands is more timer-driven as gameplay-driven since bots are implemented using a main loop, executing the same operations at each iteration, and being delayed by timers or timeouts based on data and decisions of the past. A high latency does not affect the bot's behavior, a human however adapts to the game pace involuntarily.

Since bots become never tired and react much faster than humans, in an MMORPG a character can be represented by a bot which performs e.g. a typical *kill, loot and trade* cycle (kill monsters until the bag is full, then sell all and return to a populated zone). Thus the cumbersome *up-leveling* of game characters (acquiring money and skills) can be done without human intervention. Actually a bot is only a script and can be implemented as standalone application fully independent from the MMORPG's original client. Nevertheless bots can have a graphical user interface showing the actual world map, other entities and the own character's state.

Albeit the fact that bots can affect other (human) player's fun by contaminating a game server and therefore debalancing the game, a new use case appropriately called *avatar hosting* might make sense. A currently active computer can host many bots, each initialized with the state of a remote user's avatar. The modified avatar is then implemented as a special bot with possibly individual artificial intelligence, being capable of acting autonomously in the game world. The avatar-owner saves energy while the avatar's up-leveling is outsourced. In turn the avatar-hosting computer will be rewarded for the work done, for instance by being paid with valuable items from the MMORPG, and sends back the state of the advanced avatar to the owner. Thus the avatar-owner retakes the control. The aggregation of avatars as enriched bots running on a single computer saves time and energy for many avatar-owners.

Assuming a computer's shared down/up bandwidth is 4000/500 Kbit/s (i.e., 30/3.75 Mbytes/min) and an avatar needing at most 8620/1115 bytes/min (see Table 1) this computer can host up to about 3300 avatars. Even if this computer hosts only 100 avatars, then potentially 99 other computers can reside in a low power mode. Taking into account that no further tasks run on a computer to provide the best performance for the heavy-loaded original client application while playing only for up-leveling, then resources (and energy) are wasted in comparison to the lightweight avatar-bot. Aggregation of avatars can save energy by taking over this routine-job without strong resource allocation.

Similar to video encoding we define W_A and W_B as the power consumptions of two corresponding computers in watt, R as the available (left over for the particular avatar) download bandwidth of home B, T as the average period the computer is active for the game (e.g. up-leveling for 3 hours), S as the size of the bot software that must be downloaded by B from a remote web server, and $E_A = TW_A$ joule as the energy consumption of computer A when running the bot locally. If computer B hosts an

Game	Bytes/min Down	Bytes/min Up
Everquest 2	18033	3405
EVE Online	8620	1115
World of Warecraft	15490	3814
Second Life	814909	122253

Table 1: Bandwidth demands of popular MMORPGs.

avatar for computer A, the total energy consumption therefore is

$$E_B = \frac{S}{R} W_B + T W_B \text{ joule.}$$

Similar to Sect. 5 energy can be saved if $E_B < E_A$. Assuming a fixed T and again $W_B = \beta W_A$ this yields

$$\frac{\beta S}{T(1-\beta)} < R. \tag{6}$$

For β near 1 computer B consumes almost as much power as computer A, and thus more bandwidth must be *available*, i.e., left over from other applications or avatars, to save energy. Results for $S = 10\,\text{MB}$ and $T = 3, 6$, and $9\,\text{h}$ are shown in Fig. 7. Again the critical point is at $\beta = 1$, implying $0 < \beta < 1$. However, this time the bandwidth requirements are much more moderate when compared to remote video encoding, and even today's access networks can easily be used for saving energy to a large extent.

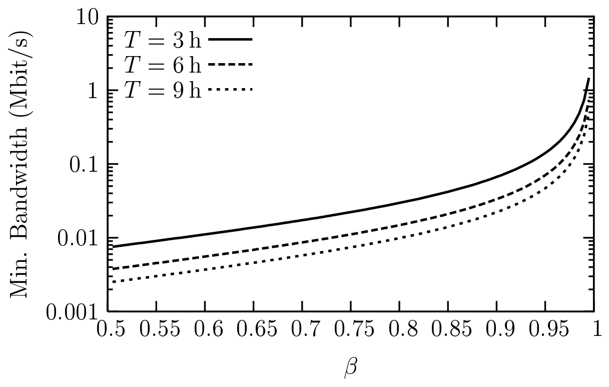

Figure 7: Minimum bandwidth required for saving energy.

As was stated above, computer B may easily host many more than just one avatar for another computer. For assessing the network costs of one avatar we measured the traffic produced by the games *Everquest 2*, *EVE Online*, *World of Warcraft* and *Second Life* as listed in Table 1 with NetLimiter's *NetLimiter 2 Monitor*[7] under tutorial conditions (no massive crowds of entities, only standard situations) which gives us an idea about the amount of data transferred during a normal human session on average.

Generally the downstream bandwidth is several times larger than the upstream bandwidth; the down/up ratio varies from 4 up to 7.7 for these measurements.

The client-side bandwidth usage of *Everquest 2* was investigated by [9] during 30 minutes sessions in a medium populated and a heavy busy zone of the game world

[7]http://www.netlimiter.com/

(realm). In case of activity, the average down/up bandwidth was 0.9/0.4 Kbit/s whereas for the busy zone 1.5/0.9 Kbit/s. Further it was discovered that *Everquest 2* does not use a bigger packet size in burst situations, only the rate of packets varies in time.

In [17] the server-sided traffic was measured for the MMORPG *Lineage II*. This study reveals the asymmetry between down/up bandwidth used depending on the number of concurrent users. Indeed this coincides with our own measurements at the client-side. The authors of [17] report that the observed downstream bandwidth was larger than the upstream bandwidth by a factor of 15, and also describe a strong linearity between the number of users and the observed bandwidth.

[8] investigates the MMORPG *Legend of Mir* and exposes the typical tiny TCP-packets often with larger payload as header size. Furthermore, client packets are smaller (mostly 19 bytes) than server packets (mostly 18 to 36 bytes) and server's mean packet interval time (375 ms) is slightly smaller than on client-side (396 ms).

In [23] server- and client-generated traffic for the MMORPG *World of Legend*, measured within a testbed for emulated GPRS/EGPRS and ADSL, was analyzed. The authors found that the average bandwidth requirements on the client-side is low enough for mobile networks, only the high RTT badly influences the game performance. The access network conditions have larger impact on the server-generated traffic as that of the client. TCP overhead and ACKs occupy a major part of the bandwidth and this is an open optimization issue. The reported results show that ADSL's performed best with respect to server- and client-generated traffic and RTT (1448 bit/s, 200 bit/s, 53.9 ms). EGPRS was slightly worse, except for the remarkable high RTT (1336 bit/s, 168 bit/s, 442.8 ms) whereas GPRS offers poorest performance (664 bit/s, 152 bit/s, 829.6 ms). A further finding shows that for this game, the traffic's packet interarrival times and packet lengths fit well with the extreme value distribution. The payload length of an uniform uplink packet on client-side is 19 bytes, but with headers over 73 bytes.

As a conclusion, avatar hosting exposes high potential for saving energy, since high aggregation factors of dozens or even hundreds of hosted avatars are possible, due to only small bandwidth requirements. Bot CPU requirements are still under investigation, however, they need much less computing power than the original game clients, since no GUI is necessary.

8 Home Management

Smart home management relies on sensors, actuators, and services which control future homes with utmost autonomy. These services will be to some extend semi-automated including user notification, e.g., ubiquitously and remotely on a mobile smart phone or locally visualized on wall or furniture integrated displays at home. For example, notifications may signal unauthorized intrusion to the home or food shortages (e.g., monitored by a smart refrigerator based on RFID technology which allows the refrigerator computing unit to determine food expiration dates as well as shortages).

Besides remote and local control of the home, sensing a variety of different home conditions is a major characteristic of smart homes. Thus, this section will focus on the sensing aspect of home management. From a networking perspective, each sensor will typically cause periodical but low constant traffic [6]. The actual message payload of sensor information is only a few bytes[8], the whole message in the range of kilobytes. The sampling and sending frequency depends on the type of sensors and the services aggregating the sensor values. Table 2 shows example sensors and the estimated load

[8]See, for example, the typical message structure of a TinyOS message consisting of ADC readings presented by http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson6.html.

Sensor type	Samples/min (estimated)	Bytes/min sent (estimated)
Temperature, wind, humidity	4	5120
Indoor position sensing	60	76800
RFID-based refrigerator	1/1440	0.89
Overall home		81921

Table 2: Example sensor sampling/sending rates for home automation services.

caused by these sensors (when assuming a UDP message of size 1280 bytes for delivering the sensor readings to a networked computer running the home management application). For indoor movement, we assume a movement speed of 1m/s and an position accuracy of one meter. We further assume that a refrigerator sends once a day an update of, e.g., expired food.

Thus, the example home would approximately cause average sensor traffic of about 81921 bytes/min ($\approx$ 10922 bit/s). When assuming a real transfer rate[9] of 4 Mbit/s, each PC can deal with sensor data sent by about $M = 366$ homes. Depending on the services' requirements, e.g., in terms of real-time responses and processing time, this number will have to be decreased and the CPU is likely to be the bottleneck (which is up to future investigations). Thus, it is possible to save energy for a number up to 365 PCs (per computer) by distributed home management based on structured P2P overlays.

Assume N homes, and each home is able to manage up to M homes. Then the number of active computers A_0 is calculated as (without considering failures):

$$A_0 = \left\lceil \frac{N}{M} \right\rceil .$$

Since home management targets embedded computing resulting in physical effects for humans, availability is a major concern and will now be included into the model. The fault-hypothesis for the home management includes only one kind of failures, that are, offline peers (crash failures), caused either by *intentionally leaves* or *peer failures.* These failures, usually referred to as peer churns, are modeled by defining two random variables X_{on} and X_{off} for a peer's online and offline time in accordance to [3] and their expectation values T_{on} and T_{off} (e.g., assuming an exponential distribution of these variables). The home management system is available if all homes can be served (all home services are available).

The corresponding error rate for a computer is $\lambda = 1/T_{on}$ and the corresponding repair rate (meaning a computer joins the P2P system) is $\mu = 1/T_{off}$. The availability can be enhanced by redundant execution of services on k different peers to tolerate up to $k-1$ failures where each service receives the sensor values by means of push information dissemination. To simplify the model, it is assumed that the services supported by one computer are replicated as a group on a disjunct set of other computers. Thus, these computers must be considered as being of the *same type* (Types 1 to A_0 as shown in Figure 8). In the faultless state the system consists of A_0k computers. Figure 8 depicts the birth-death process of the fault-tolerant system.

Each (computer) failure changes the system state from 0 to $(k-1)A_0 + 1$ (where until $k-1$ failures the system is in any case available for all homes, within the range of k to $A_0(k-1)$ the system availability depends on which computers fail in parallel and in state $A_0(k-1)+1$ the system is no longer available (i.e., at least M services are not available any more).

[9]Here, 1 Mbit = 10^6 bit

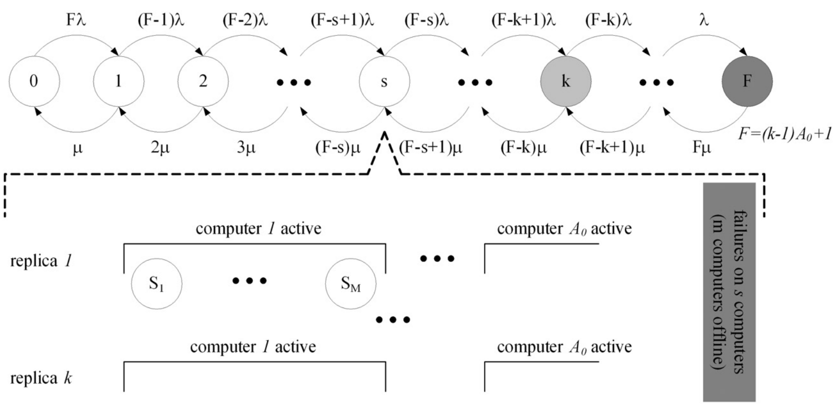

Figure 8: Failures modeled as birth-death process for home management.

For the states i where $k \leq i \leq A_0(k-1)$, the probability that the system is available can be calculated using the hypergeometric distribution[10] to find the number of occurrences of a set of interesting jk failures among the i failures (i is here the number of the sample set and jk is both the number of occurrences in the basic set A_0k, and the number of occurrences in the sample set). Due to multiple inclusion of cases, the inclusion-exclusion principle has to be applied resulting in the following calculation for the availability in state i:

$$Av_i = \begin{cases} 1, & 0 \leq i \leq k-1 \\ 1 - \sum_{j=1}^{j=\lfloor i/k \rfloor} (-1)^{j+1} \dfrac{\begin{pmatrix} jk \\ jk \end{pmatrix} \begin{pmatrix} A_0k - jk \\ i - jk \end{pmatrix}}{\begin{pmatrix} A_0k \\ i \end{pmatrix}} \begin{pmatrix} A_0 \\ j \end{pmatrix}, & k \leq i \leq A_0(k-1) \\ 0, & i > A_0(k-1). \end{cases}$$

The probabilities for the system being in one of the states i is given by p_i. The probabilities of the system's steady state are given as follows [4], where $\sum_{i=0}^{(k-1)A_0+1} p_i = 1$:

$$p_0 = \frac{1}{\sum_{i=0}^{A_0(k-1)+1} \begin{pmatrix} A_0(k-1)+1 \\ i \end{pmatrix} \left(\frac{\lambda}{\mu}\right)^i}, \text{ and}$$

$$p_i = \begin{pmatrix} A_0(k-1)+1 \\ i \end{pmatrix} \left(\frac{\lambda}{\mu}\right)^i p_0, \quad \text{for} \quad 0 \leq i \leq A_0(k-1)+1.$$

The availability of the system can, thus be calculated as:

$$Availability = \sum_{i=0}^{A_0(k-1)} p_i Av_i.$$

The overall number A of active computers necessary to maintain home services with a configurable redundancy value k is calculated by considering the probabilities of the

[10]For a quick reference see http://en.wikipedia.org/wiki/Hypergeometric_distribution

system being in one of these faulty states:

$$A = \left\lceil \sum_{i=0}^{(k-1)A_0+1} p_i \left(A_0 k - i\right) \right\rceil .$$

Figure 9 shows the active computers necessary to support up to 1000 sensor-enhanced homes (as specified in this section) considering $k = 3$ redundant service executions, under different configurations of the maximum load possible at the modeled home computers ($M = 50, 100, 200$) and different failure rates in accordance to the assumptions and findings in [3] ($\mu = 1/10, 1/15$, $\lambda = 1/15, 1/10$). Without sharing, each home would run each own controlling computer providing less availability ($k = 1$) by consuming more energy. The energy efficiency ν can be calculated as the number of homes served (in our model equal to home services supported) per kWh consumed by the active computers. The trends show that in addition to increasing the possible load per computer, also higher error rates and lower repair rates can beneficially influence energy saving. The latter will lead to a malign decrease in the system's availability. Thus, both energy saving and achievable availability have to be investigated to derive a trade-off depending on the services' requirements. The availability achievable is depicted in Figure 10.

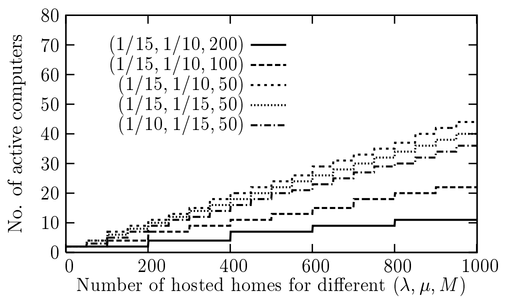

Figure 9: Home management: Active computers.

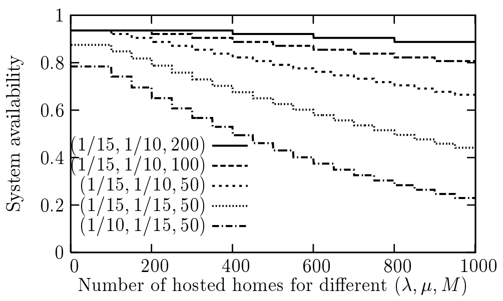

Figure 10: Home management: Achieved availability.

9 Conclusion

In this paper we propose to aggregate computing tasks on a small number of nodes in order to save energy by hibernating as many idle computers as possible for home networks. We have shown a first draft for a distributed architecture for carrying out such an aggregation in [11]. In this paper we analyze several home application scenarios concerning their energy saving potentials. The best applications to aggregate would run on a 24/7 basis, or at least for a long time, and only require little computational or networking demands. Examples for such applications are avatar hosting and home management. An application requiring high network demands is given by download sharing, which however does offer good potential for energy saving. Yet a more difficult application is remote video encoding, for which we have analyzed under which circumstances energy can be saved.

As a conclusion, all analyzed scenarios offer various degrees of energy efficiency, and future work will focus on a simulation tool for evaluating the above described applications, as well as others in more details.

10 Acknowledgment

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] Robert D. Atkinson. The case for a national broadband policy. http://www.itif.org/files/CaseForNationalBroadbandPolicy.pdf.

[2] P. Bertoldi and B. Atanasiu. Electricity Consumption and Efficiency Trends in the Enlarged European Union. Technical report, Institute for Environment and Sustainability, European Commission Report EUR 22753 EN, 2007.

[3] A. Blinzenhoefer and K. Leibnitz. Estimating Churn in Structured P2P Networks. *Managing Traffic Performance in Converged Netwokrs*, 4516/2007, 2006.

[4] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. *Queueing Networks and Markov Chains*. Wiley & Sons, 2nd edition, 2006.

[5] K.-T. Chen, J.-W. Jiang, P. Huang, H.-H. Chu, C.-L. Lei, and W.-C. Chen. Indentifying mmorp bots: A traffic analysis approach. In *ACE 06*, June 2006.

[6] S. Ci, H. Sharif, and D. Peng. An Effective Scheme for Energy Efficiency in Mobile Wireless Sensor Networks. In *2004 IEEE INternational Confernece on Communications*, pages 3468–3490, 2004.

[7] J.A. Clarke, C.M. Johnstone, P.A. Strachan, and J. Kim. Client-Centered, Energy-Efficient Wireless Communication on IEEE 802.11b Networks. *Energy and Buildings, Elsevier*, Vol 36, August 2004.

[8] L. Fang, Y. Guotao, and Z. Wenli. Traffic recognition and characterization analysis of mmorpg. In *International Conference on Communication Technology Proceedings (ICCT)*, 2006.

[9] Fritsch, Ritter, and Schiller. The effect of latency and network limitations on mmorpgs. In *NetGames'05*, October 2005.

[10] Tom's Hardware. Desktop cpu charts - encoding video xvid. http://www.tomshardware.com/de/charts/desktop-cpu-charts/encoding-video-xvid,356.html.

[11] H. Hlavacs, K.A. Hummel, R. Weidlich, A. Houyou, A. Berl, and H. de Meer. Energy efficiency in future home environments: A distributed approach. In *IFIP TC6's and IEEE's 1 st Home Networking Conference, Paris, France*, December 2007.

[12] IBM Virtualizatin View. Virtualization Can Help Power Efficiency. http://www-03.ibm.com/systems/virtualization/view/011607.html, January 2007.

[13] Intel. Energy Star* System Implementation. www.intel.com/cd/channel/reseller /asmo-na/eng/339085.htm, 2007.

[14] 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.

[15] C.E. Jones, K.M. Sivalingam, P. Agrawal, and J.C. Chen. A survey of energy efficient networkprotocols for wireless networks. *Wireless Networks*, Vol 7(Issue 4), 2001.

[16] S. Karatasou, V. Geros, and M. Santamouris. On the potential of Internet based energy services in Greece during cooling season. In *International Conference Passive and Low Energy Cooling for the Built Environment*, May 2005.

[17] Kim, Choi, Chang, Kwon, Choi, and Yuk. Traffic characteristics of a massively multi-player online role playing game. In *NetGames'05*, 2007.

[18] J.G. Koomey, editor. *Energystar, Server Energy Measurement Protocol, Version 1.0*, Following Energy Efficiency Server Benchmark Technical Workshop, Santa Clara, CA, March 2006.

[19] 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.

[20] L. Peterson and T. Roscoe. The Design Principles of PlanetLab. *ACM SIGOPS Operating Systems Review*, Vol 40(Issue 1):11–16, 2006.

[21] S. Rivoire, M. Shah, P. Ranganathan, and C. Kozyrakis. JouleSort: A Balanced Energy-Efficiency Benchmark. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, June 2007.

[22] C. Windeck. Energy Star 4.0. *C't German Magazine for Computer Techniques*, Vol 14:Pages 52–53, 2007.

[23] Yi Wu, Hui Huang, and Dongmei Zhang. Study the Traffic Difference of Online Games Between GPRS/EGPRS and ADSL Networks. *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, Volume 2:Pages 1011 – 1016, 2006.

[24] Haijin Yan, S.A. Watterson, D.K. Lowenthal, Kang Li, R. Krishnan, and L.L. Peterson. Client-Centered, Energy-Efficient Wireless Communication on IEEE 802.11b Networks. *IEEE Transactions on Mobile Computing*, Vol 5(Issue 11), 2006.