Design and Evaluation of a BitTorrent Proxy for Energy Saving

Giuseppe Anastasi[*], Marco Conti[#], Ilaria Giannetti[*], Andrea Passarella[#]

[*] *Dept. of Information Engineering*
University of Pisa, Italy
{firstname.lastname}@iet.unipi.it

[#]*CNR-IIT*
National Research Council, Italy
{firstname.lastname}@iit.cnr.it

Abstract

Recent studies indicate that the Internet-related energy consumption represents a significant, and increasing, part of the overall energy consumption of our society. The largest contribution to this consumption is due to Internet edge devices. Typically, users leave their PCs continuously on for satisfying the connectivity requirements of file sharing p2p applications, like BitTorrent. In this paper we propose a novel proxy-based BitTorrent architecture. BitTorrent users can delegate the download operations to the proxy and then power off, while the proxy downloads the requested files. We implemented our solution and validated it in a realistic testbed. Experimental results show that, with respect to a legacy approach, our solution is very effective in reducing the energy consumption (up to 95%) without introducing any QoS degradation.

1. Introduction

In the last years, Internet-related energy consumption is becoming one of the major research issues for the networking community. Several reports show that the Internet energy consumption is already too high and, without paying attention to it, the problem will become more and more relevant while the Internet role in the society expands. As reported in [6] about 74 TeraWatts hours (TWh) per year of electricity are consumed by Internet and, although this is only the 2% of the global energy consumption in USA, it's a considerable number. Furthermore, it is estimated that by adopting power management techniques on the Internet devices, a 32% energy saving can be achieved. These numbers have stimulated efforts to reduce the Internet energy consumptions. Researchers' efforts tend to concentrate on the network edges (i.e., data-center networking equipment [4] or personal computing devices), as there is no much to save inside the Internet core [9]. In this paper we focus on energy management in personal computing device (PCs) as they are widespread and very numerous. Furthermore, they are often managed by common users who, typically, do not pay so much attention to the energy problem (e.g., they often leave their PC always on). Indeed, as reported in [16], during 2007 in USA data centers accounted for approximately 2 TWh per year, while office and home devices accounted for approximately 16 TWh per year. Furthermore, users generally do not apply any power-saving policy. This clearly emerges, for example, in the PC Energy Report by the UK National Energy Foundation [12] related to the energy consumption of PCs used at work. This report highlights that about 21% of PCs used at work are almost never switched off (during nights and weekends), and this causes energy wastage of about 1.5 TWh of electricity per year, corresponding to about 700,000 tons of CO2. However, energy wastage due PCs left always on, for laziness or omission, is only a part of the problem and could be easily avoided by means of commercially available devices (e.g., NightWatchman [12]) can perform a centralized shutdown of all PCs at a predefined time. Instead, the most interesting case is when a PC remains continuously powered on to perform network activities like, for example, running a p2p file-sharing application. Recent studies [14] indicate that 40-73% of Internet traffic is p2p, and BitTorrent is the most popular p2p protocol (about of 50-75% of the overall p2p traffic).

Based on these remarks, in this paper we focus on policies for saving energy in PCs running a p2p application. Specifically, our solution is targeted to PCs using the BitTorrent platform. However the ideas and concepts presented here can be easily extended to other p2p platforms as well.

Traditional power management techniques [2] that power off the network interface when the PC is not using the network, are inadequate in an environment where permanent connections are required. In the literature we can identify three different categories of power management techniques compatible with permanent connectivity: *adaptive link rate*, switching between different *power management levels*, and *proxy-based* solutions. Techniques based on adaptive link rate rely on the evidence that the energy consumption of the Network Interface Card (NIC)

strongly depends on the supported link rate. For example, the power consumption of Ethernet NIC increases from 1W for 10/100 Mb/s, to 7W for 1 Gb/s, and up to 15 W for 10 Gb/s. The basic idea of adaptive link rate is, thus, to adjust the link rate according to the real needs. The idea is known as Adaptive Link Rate (ALR) [9] or Rapid PHY Selection (RPS) [7]. Techniques based on switching between different power management levels are targeted to NICs with different power modes (from completely sleeping to completely active). They switch the NIC from one mode to another, depending on the network activity, e.g., as in Dynamic Ethernet Link Shutdown (DELS) [10, 11]. While these two techniques can provide some energy saving, they do not seem to be the best approach for our environment where a file download can last for several hours. In this case, we believe that delegating the download operations to a proxy server, and shutting down the PC during the download phase, is the most effective solution. Possibly, the proxy server could be running on a computer providing other network services, (e.g., DHCP server, DNS etc.).

Proxy-based architectures are not new, and have been already considered for energy-efficient Internet access from mobile devices. However, in that case, the proxy architecture was designed for supporting a mobile device running legacy client-server applications [1]. More recently, the idea of a proxy-based architecture has been proposed for implementing energy-aware solutions in the fixed Internet (e.g., [8]). The idea is to use a proxy that takes the host place to respond to minimal network interactions and wakeup the host only when the network requires the host interaction. In this case, a wakeup mechanism that awakes the host is required (e.g., the Wake On LAN NIC [5]). The solution presented in [8] provides a general framework for saving the energy consumed by the NIC, and is not tailored to any specific p2p platform. Instead, our solution introduces a p2p energy-aware platform that makes possible to completely shut down the client PC. It is worth noting that the wakeup mechanism might be integrated in our architecture for waking up the PC as soon as the proxy has completed the download operation. However, p2p file-sharing applications, generally, do not require that the downloaded file is immediately transferred from the proxy to the client PC. This can be done at a later time, e.g., when the user connects again to the Internet.

In the next section we will present and discuss our solution for saving energy when using BitTorrent for downloading files, while in Section 3 we will present an experimental evaluation of our proposal.

2. BitTorrent Energy-saving Architecture

Before describing the proposed BitTorrent energy-saving architecture, we provide a brief overview of the standard BitTorrent architecture. More details can be found in [3].

BitTorrent implements an unstructured overlay network customized for file sharing. Nodes of the overlay are called *peers*. The basic idea of BitTorrent is that peers both download and upload *parts* of the shared files. This results in the fact that each peer downloads a given file from a multitude of other peers, instead of downloading it from a single server as in a conventional client-server model. The resulting capacity of such cooperative downloading process is higher than that of the traditional client-server architectures [13]. A tagged peer wishing to download a file from scratch needs to get a corresponding *torrent* file (hereafter referred to as torrent). Torrents are very small, are typically hosted by conventional Web servers, and can be found through standard Internet search engines. A torrent contains the name of the file's *tracker*, with whom the tagged peer connects first. The tracker is a node that constantly tracks which peers have parts of the file. The tagged peer receives from the tracker a random list of peers, that the tagged peer can contact to start the download process. Peers participating in the download of the same file are collectively called a *swarm*. At any point in time a peer in a swarm is in touch with a set of *neighbors* with which it exchanges parts of the file. The neighbor set dynamically changes, mainly according to the "Tit-for-Tat" (TAT) policy. Each peer preferentially selects – for uploading parts of the shared files – those neighbors from which it can download at the highest rate. Once every 30 seconds neighbors are selected completely at random, as a way to discover new neighbors, and allow new peers in a swarm to start-up. Finally, each peer downloads from neighbors the parts according the Rarest First policy (i.e., parts which are less spread are downloaded first).

2.1. Energy-saving BitTorrent

The legacy BitTorrent architecture is not "energy-friendly". BitTorrent peers have to stay connected to the overlay network during the whole download of requested files, which may typically take several hours. Periodically turning off peers without modifying the BitTorrent architecture is not a viable solution for several reasons. If a peer is downloading content, powering it off does not save any energy, as the download itself stops when the peer turns off. Also, powering off peers that are not downloading anything (but are sharing content) is not an efficient solution in

general, as this can result in decreasing the overall download performance of the swarms they participate to. Thinking at coordinated ways of powering those peers is also not appropriate, as would require central control, and is at odds with the BitTorrent P2P design paradigm.

In this paper we propose a proxy-based energy-saving architecture to overcome these drawbacks. We assume a standard LAN environment where several users run BitTorrent peers on their PC. We pick one PC in the LAN to behave as a proxy between the peers and the rest of the BitTorrent network. The proxy can be either a dedicated PC, or a PC that has to be continuously powered on for other reasons (the latter is the best case from an energy saving standpoint). The basic idea is that peers "behind" the proxy ask the proxy itself to download the requested content on behalf of them. The proxy participates to the conventional BitTorrent overlay, and takes care of all the downloads of the peers behind it. While downloads are in progress, the peers behind the proxy can be switched off. The requested files are then transferred from the proxy to the peers upon completion.

This architectural design is clearly suitable to save energy, and also keeps the underlying P2P principles of the original BitTorrent architecture. The BitTorrent network is not modified, as the proxy acts exactly as a standard BitTorrent peer. Modifications are just required at the proxy and the clients behind the proxy, and are thus confined within a single LAN. Note that different proxies "masking" peers on different LANs are completely independent of each other. Therefore, this architecture is also scalable, as it does not require modifications of the BitTorrent global architecture, nor global coordination between (sets of) BitTorrent peers. Finally, this architecture is also suitable to support mobile clients accessing the Internet and, more in general, is a solution to enable *asynchronous BitTorrent downloads*, which is something not supported by the conventional BitTorrent architecture.

2.2. Architecture and Protocols

The proposed architecture falls in the family of traditional split architectures, e.g. [15]. The architectural components between a peer and the proxy are shown in Figure 1. The BT peer at the proxy is a standard BitTorrent peer. This peer is in charge of downloading the contents requested by all users behind the proxy. In the "internal" part of the architecture (i.e., between the proxy and the user's PC), we adopt a simple client/server scheme implemented by the *Energy-Saving BitTorrent* (ESBT) modules at the user PC and proxy (see Figure 1).

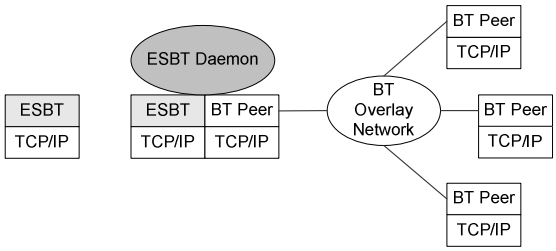

Figure 1. Energy-saving BitTorrent architecture.

The ESBT module at the proxy continuously monitors incoming requests for new downloads coming from the clients, hands them over to the *ESBT Daemon*, which translates them in download requests issued by the BT peer running on the BitTorrent overlay network. Besides requests for new files, clients can also issue requests for checking the status of previously requested files, as well as requests to fetch files from the proxy, when downloads are complete. Between any successive requests, clients can be turned off.

Finally, clients can also upload content to the proxy, that has to be shared on the BitTorrent overlay. This is also an important advantage of our architecture. The BT peer at the proxy can share all the files that would be shared by individual BT peers running on the user PCs. Therefore the BT peer at the proxy is likely to receive more download bandwidth than any individual BT peer (in the case no proxy is used). Thus, our proxy-based architecture can also achieve lower download times for all users. We provide preliminary results showing this feature in Section 3.

The proposed architecture requires very simple networking protocols. When a user wishes to download a new file, the ESBT client running on the user PC retrieves the .torrent file from a torrent server in the Internet, as in the conventional BitTorrent architecture. Then, it uploads the .torrent file to the ESBT server running on the proxy. The server hands over this file to its BT peer to start the download. Upon receiving an ACK from the server, indicating that the download has successfully started, the client records that the download is ongoing, and the user PC can be turned off. When the client on the user PC is restarted, it checks which downloads it has previously requested, for each of them it asks the server for a status update. If the file's download is completed, it downloads the file from the server.

3. Performance Evaluation

To analyze the effectiveness of our proxy-based BitTorrent architecture we performed several real experiments. Specifically, the main objective of the proposed system is to maximize the energy saving, with

respect to the legacy approach, without introducing a significant degradation of the Quality of Service (in terms of file download time). In this section, we present the details of our experimental testbed, the metrics used to quantify the effectiveness of our approach, and the experimental results we obtained.

The experimental environment is based on a set of PCs connected to a Gigabit Ethernet LAN and, through this network, they are interconnected to the Internet via a high-speed 100 Mbps link. By exploiting the set of PCs we implemented two systems: a legacy BitTorrent system and one based on the BitTorrent proxy we have developed. All the PCs use Linux Ubuntu 8.04, and the BitTorrent client is a light command-line client provided with Rasterbar libtorrent.

By exploiting the two systems, we performed a large set of experiments to measure their performance when downloading the same set of files. More precisely, for each experiment we identified a given number, *n*, of files to download and we assigned one download operation per PC. The same experiment was repeated several times with the same number n of files but changing the set of files. To have comparable statistics we selected files that are approximately of the same size (they are in the range [3.95 GB, 4.71 GB]), have similar popularity and, for each file, the initial number of *seeds* (i.e., peers that already have the whole file) is in the range [200, 800]. To have similar experimental conditions, all the experiments are interleaved, so that the compared results are obtained with similar congestion conditions of the Internet, and a similar number of peers[1].

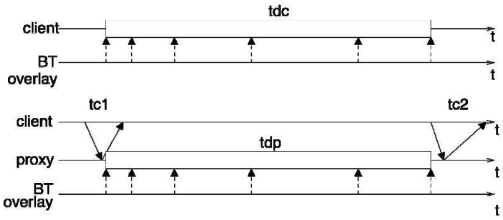

Figure 2. Time that the client is powered ON: without proxy (top) and with proxy (bottom).

To evaluate the effectiveness of our proxy-based architecture, we introduce a set of performance indices. As shown in Figure 2, we denote by t_{dc} and t_{dp} the file download time when using the legacy and our proxy-based architecture, respectively. When using the proxy, we have to consider two additional delays: t_{c1} is the time for delegating the download operation to the proxy, while t_{c2} is the time taken by the client to fetch the file from the proxy. According to these definitions, any PC running the BitTorrent client must be powered on for at least $t_{withP} = t_{c1} + t_{c2}$ time units when using the proxy, and $t_{withoutP} = t_{dc}$ when using the legacy architecture. Since the energy consumption is proportional to the time the PC running the BitTorrent client is powered on, for ease of reading hereafter we assume that we consume an energy unit for each unit of time the PC is powered on, and hence the total energy consumption is exactly equal to the total time the PC must be powered on. We can now introduce the following energy saving index:

$$I_{saving} = 1 - \frac{t_{withP}}{t_{withoutP}} = 1 - \frac{t_{c1} + t_{c2}}{t_{dc}}$$

Specifically, I_{saving} is the percentage of time an individual PC can be turned off with respect to the downloading time without using the proxy. As we assume that the time is proportional to the energy, I_{saving} also denotes the percentage of energy saving for the client PC. It refers to the case where the proxy runs on a PC already continuously powered on for other purposes. When the proxy runs on a dedicated PC, we have to introduce a different index taking into account the energy consumption of the proxy, defined as:

$$I^*_{saving} = 1 - \frac{t_{withP} + t_{dp}}{t_{withoutP}} = 1 - \frac{t_{c1} + t_{c2} + t_{dp}}{t_{dc}}$$

It is clear that, as $t_{dp} \approx t_{dc}$[2] holds, $I^*_{saving} < 0$, i.e., the energy spent when a single PC uses the proxy architecture is higher than the energy of the legacy architecture. This is of course expected. However, we can expect an energy saving if more PCs utilize, at the same time, the BitTorrent proxy for downloading several files in parallel. Let us generalize the energy saving indices I_{saving} and I^*_{saving} to the case when n PCs, each running a BitTorrent client, download a file in parallel. By denoting with $I_{saving}(n)$ and $I^*_{saving}(n)$ the energy saving indices as a function of the number of clients n we obtain:

$$I_{saving}(n) = 1 - \frac{\sum_{i=1}^{n} t_{withP}(i)}{\sum_{i=1}^{n} t_{withoutP}(i)} \qquad I^*_{saving}(n) = 1 - \frac{t_{dp} + \sum_{i=1}^{n} t_{withP}(i)}{\sum_{i=1}^{n} t_{withoutP}(i)}$$

where $t_{withP}(i)$ and $t_{withoutP}(i)$ is the total time the i-th client must be powered on, with or without the

[1] Both the Internet conditions and the number of peers interested to a file are not under our control. By interleaving the experiments with and without the proxy we have been able to limit the variability of these parameters between successive experiments.

[2] The experimental results will confirm that the time to download a file with or without the proxy is almost the same. Indeed, generally, using the proxy the download delay reduces.

BitTorrent proxy, respectively, and t_{dp} is the total time the proxy must be powered on for completing the download of all n files (which can be approximately assumed equal to the time to download a single file, as the downloads are in parallel, and the file sizes and popularities are similar). By considering the above indices, in our experimental scenario we get the results presented in Figure 3.

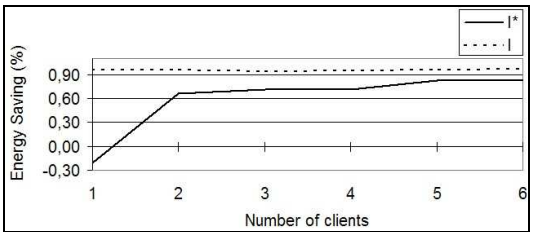

Figure 3. Energy saving vs. number of clients.

These results show that, if we do not consider the energy consumption of the BitTorrent proxy, the percentage of energy saving does not depend on the number of clients and is approximately 95% for each client. On the other hand, when we consider the proxy energy consumption, the percentage energy saving increases with the number of files to download (as the proxy cost is subdivided between an increasing number of files) and asymptotically converges to $I_{saving}(n)$. These results can be easily explained. By assuming that (i) all files have approximately the same download time t_{dc}, (ii) all clients experience the same delay for uploading the request to the BitTorrent proxy and downloading the file from the proxy (i.e., t_{c1} and t_{c2}), and (iii) the interference among clients is negligible due to the gigabit bandwidth of the LAN, it yields

$$I_{saving}(n) \approx 1 - \frac{n \cdot (t_{c1} + t_{c2})}{n \cdot t_{dc}} = 1 - \frac{(t_{c1} + t_{c2})}{t_{dc}} = I_{saving}(1)$$

This means that, if the proxy runs on a PC already continuously powered on for other purposes, the percentage energy saving does not depend on the number of files to download. This also suggests that the absolute energy saving is (approximately) linearly increasing with the number of clients. To show this we introduce the $E_{saving}(n)$ and $E^{*}_{saving}(n)$ indices, which measure the total time the client PCs can be powered off when using the proxy architecture with respect to the legacy case (no proxy). As above, $E^{*}_{saving}(n)$ refers to the case when a dedicated PC is used as proxy.

$$E_{saving}(n) = \sum_{i=1}^{n} t_{withoutP}(i) - \sum_{i=1}^{n} t_{withP}(i)$$

$$E^{*}_{saving}(n) = \sum_{i=1}^{n} t_{withoutP}(i) - \left(t_{dp} + \sum_{i=1}^{n} t_{withP}(i) \right)$$

By measuring $E_{saving}(n)$ and $E^{*}_{saving}(n)$ in our experiments we obtain the results plotted in Figure 4. These results clearly indicate that in both cases the overall energy saving increases with the number of files and the difference between the two curves is only due to the proxy energy consumption. Again, it is possible to provide an analytical explanation for the behaviors presented in Figure 4.

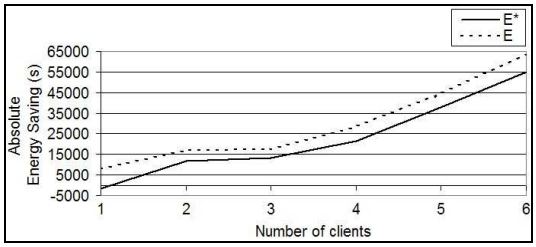

Figure 4. Absolute energy saving at clients expressed as corresponding time.

Specifically, by following the same arguments used above, it is easy to observe that $E_{saving}(n)$ increases almost linearly with the number of files:

$$E_{saving}(n) \approx n \cdot [t_{dc} - (t_{c1} + t_{c2})] \approx n \cdot E_{saving}(1)$$

On the other hand, when we include the energy consumed by the proxy in the total energy consumption, we have

$$E^{*}_{saving}(n) \approx n \cdot [t_{dc} - (t_{c1} + t_{c2})] - t_{dp}$$

Hence by assuming, as before, $t_{dc} \approx t_{dp}$ it follows that:

$$E^{*}_{saving}(n) \approx (n-1) \cdot t_{dc} - n \cdot (t_{c1} + t_{c2}) \xrightarrow{n} E_{saving}(n)$$

Furthermore, by following the same line of reasoning we have:

$$I^{*}_{saving}(n) \approx 1 - \frac{n \cdot (t_{c1} + t_{c2}) + t_{dp}}{n \cdot t_{dc}} \approx 1 - \frac{(t_{c1} + t_{c2})}{t_{dc}} - \frac{1}{n}$$

$$\xrightarrow{n} 1 - \frac{(t_{c1} + t_{c2})}{t_{dc}} = I_{saving}(n)$$

The two formulas above indicate that, for large n, the energy consumption of the BitTorrent proxy can be neglected. The results presented above clearly show the effectiveness of the proxy-based architecture from the energy-saving standpoint. The other aspect that we need to investigate is the impact of the proxy-based architecture on the download time. In the previous analysis we have assumed that the time to download a file is not significantly affected by the proxy. To analyze this aspect, in our experiments we also compare the time to download n files in parallel with, and without, the proxy. The results of this analysis are summarized in Figure 5, where we plot the average download time of a single file, for an increasing number of parallel downloads (each column represents the average of several experiments).

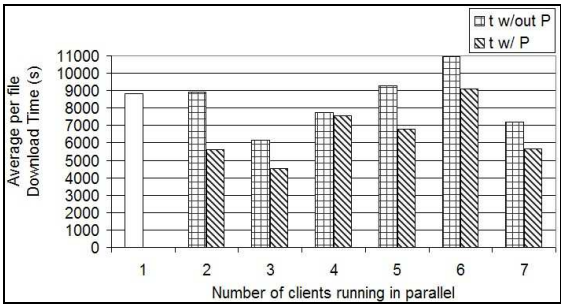

Figure 5. Average per-file download time.

We tried to mitigate the network variability by replicating the experiments several time, but it is very difficult (if not impossible) to remove the effects of this variability. In any case, the results reported in the figure clearly indicate that using the BitTorrent proxy does not introduce any degradation in the QoS; indeed, on average, the time to download a file *reduces* by exploiting the proxy architecture. This can be explained by taking into account that the BT peer on the proxy shares more files on the BT overlay with respect to any single BT peer in the legacy architecture, and thus gets higher download bandwidth. To quantify the average gain we can achieve, we computed the average download time over all the experiments we performed. With the BT proxy the average download time reduces by approximately 22% (6541s vs. 8439s). We wish to conclude the analysis of the delay with an interesting observation tightly coupled with the BitTorrent behavior. Specifically, we analyze how the availability of a single (popular) file to upload on the proxy can highly reduce the download time of all files the proxy is downloading. This effect is well exemplified by results presented in Figure 6 by considering the proxy architecture and comparing the delay to download 4 files in parallel, with or without a popular file available for other BitTorrent peers on the proxy.

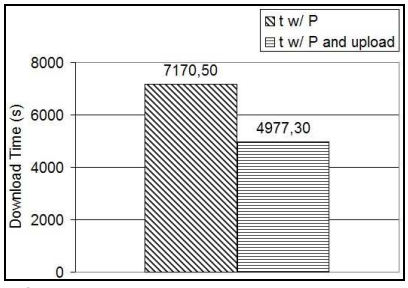

Figure 6. Effect of popularity on download time.

As it clearly appears from Figure 6, a single popular file can further reduce (with respect to the gain already achieved with the proxy) of about 25-30% the download time of all the other files. In addition to energy saving, this provides a strong motivation for exploiting our proposed architecture: a single popular file shared on the proxy provides a high benefit to everyone. Therefore, we can expect that a wise policy to reduce the downloading times is to concentrate both the files to download and the popular files to upload on the proxy. However more experimental results are required to confirm this hypothesis.

4. References

[1] G. Anastasi, M. Conti, E. Gregori, A. Passarella, "Performance Comparison of Power Saving Strategies for Mobile Web Access", *Performance Evaluation*, Vol. 53, N. 3-4, Aug 2003.

[2] G. Anastasi, M. Conti, A. Passarella, "Power Management in Mobile and Pervasive Computing Systems", *Algorithms and Protocols for Wireless and Mobile Networks*, Chap 24, Azzedine Boukerche (Editor), CRC, Oct 2005.

[3] A. R. Bharambe, C. Herley, V. N. Padmanabhan, "Analyzing and Improving BitTorrent Performance", *Technical Report MSR-TR-2005-03*, Feb 2005.

[4] L.A. Barroso, U. Holzle, "The Case for Energy-Proportional Computing," *IEEE Computer*, Vol. 40, N. 12, 2007.

[5] Christensen K., Gulledge F., "Enabling power management for network-attached computers", *International Journal of Network Management*, Vol. 8, N. 2, Mar – Apr 1998.

[6] Christensen K., George A. D., "Increasing the Energy Efficiency of the Internet with a Focus on Edge Devices", in: *The Energy Efficient Internet Project*, University of South Florida and University of Florida, Florida, 2005 – 2008.

[7] Christensen K., Blanquicet F., "An initial performance evaluation of Rapid PHY Selection for Energy Efficient Ethernet", Proc. *IEEE Conference on Local Computer Networks*, Oct 2007.

[8] Ken Christensen and Bruce Nordman, "Improving the Energy Efficiency of Ethernet-Connected Devices: A Proposal for Proxying", *Ethernet Alliance White Paper*, Sept 2007. http://efficientnetworks.lbl.gov/enet-pubs.html

[9] Greg Goth, "The Net's Going Green", *IEEE Internet Computing*, Vol. 12, N.1, January -February 2008.

[10] Gunaratne C., Christensen K., "Ethernet Adaptive Link Rate: System Design and Performance Evaluation", *Proc. IEEE Conference on Local Computer Networks*, Nov 2006.

[10] Gupta M., Grover S., Singh S., "A feasibility study for power management in LAN switches", *IEEE ICNP*, Germany, Oct 2004.

[11] Singh S., Gupta M., "Using low-power modes for energy conservation in Ethernet LANs", Proc. *INFOCOM 2007*, Portland USA, May 2007.

[12] Karayi S. (NEF), "The PC energy report", 1E, London, 2007. *www.1e.com/energycampaign/downloads/1E_reportFINAL.pdf*

[13] Don Towsley, "The Internet is Flat: A brief history of networking over the next ten years", *ACM PODC*, 2008.

[14] Schulze H., Mochalski K., "The Impact of Peer-To-Peer file sharing, voice over IP, Skype, Joost, Instant Messaging, One-Click Hosting and Media Streaming such as YouTube on the Internet", *IPOQUE – Internet Study 2007*, Leipzig, Germany, Sept 2007.

[15] G. Anastasi, M. Conti, W. Lapenna, "A Power Saving Network Architecture for Accessing the Internet from Mobile Computers: Design, Implementation and Measurements", *The Computer Journal*, Vol. 46, N. 1, Jan 2003.

[16] C. Gunaratne, K. Christensen, S. Suen, and B. Nordman, "Reducing the Energy Consumption of Ethernet with an Adaptive Link Rate", *IEEE Transactions on Computers*, V.57, N.4, Apr 2008.