Volume 33, Issue 7, 3 May 2010 ISSN 0140-3664
ELSEVIER
computer
communications
Available online at www.sciencedirect.com
ScienceDirect

Computer Communications 33 (2010) 794–802

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A BitTorrent proxy for Green Internet file sharing: Design and experimental evaluation

Giuseppe Anastasi [a], Ilaria Giannetti [a], Andrea Passarella [b,*]

[a] *Dept. of Information Engineering, University of Pisa, via Diotisalvi 2, 56122 Pisa, Italy*
[b] *IIT-CNR, via G. Moruzzi 1, 56124 Pisa, Italy*

ARTICLE INFO

Article history:
Received 22 September 2009
Received in revised form 17 November 2009
Accepted 20 November 2009
Available online 5 December 2009

Keywords:
P2P file sharing
BitTorrent
Energy efficiency
Green Internet

ABSTRACT

Recent studies have shown that the Internet-related energy consumption represents a significant, and increasing, part of the overall energy consumption of our society. Therefore, it is extremely important to look for energy-efficient Internet applications and protocols. The largest contribution to this energy consumption is due to Internet edge devices (PCs and data centers). As a particularly significant example, in this paper we address the fact that users leave their PCs continuously powered on for satisfying connectivity requirements of Peer-to-Peer (P2P) file sharing applications, like BitTorrent (currently the most popular P2P Internet platform). To reduce these energy consumptions, without penalizing the Quality of Service of BitTorrent users, in this paper we propose a novel architecture based on the introduction of a BitTorrent proxy. BitTorrent users delegate the download operations to the proxy and, then, power off their PC, 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 the legacy BitTorrent approach, our solution is very effective in reducing the energy consumption without introducing any QoS degradation. Specifically, our results show that the proxy-based solution can provide up to 95% reduction in the energy consumption and, at the same time, a significant reduction in the average file download time.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Several reports indicate that the total Internet-related energy consumption is already very high and is expected to increase even more in the next future as the Internet role in the society will expand. About 74 TeraWatts hours (TWh) per year of electricity are consumed in USA by Internet equipments [1]. Although this just accounts for 2–3% of the global electricity consumption in USA [2], it is nevertheless a remarkable number. It is estimated that about 32% of this energy could be saved by just using power management techniques on Internet-connected devices [1]. These figures have stimulated the efforts of the networking community to reduce the Internet-related energy consumption, and greening of the Internet is nowadays one of the hottest research topics.

Researchers' efforts tend to concentrate on the network edges – i.e., data centers and personal computing devices (PCs) – as there is not so much room for energy savings inside the Internet core [3]. In this paper we focus on PCs as they are widespread and very numerous (over a billion in the world [2]). In 2007 data centers in USA accounted for approximately 2 TWh per year, while office and home PCs accounted for approximately 16 TWh per year [4]. Furthermore, PCs are typically managed by common users who are not very eager to address the energy problem and, so, they often leave their PC always powered on. For example, the PC Energy Report by the UK National Energy Foundation [5] has highlighted that about 21% of the PCs used at work are almost never switched off during nights and weekends, thus causing a energy wastage of about 1.5 TWh of electricity per year (corresponding to about 700,000 tons of CO2). This energy wastage could be easily avoided by just switching off PCs, e.g., using a centralized shutdown solution such as the NightWatchman [5]. However, many PCs are intentionally left on by their users, especially at home, to perform networking activities like, for example, Peer-to-Peer (P2P) file-sharing. Recent studies [6] indicate that a very large fraction of nowadays Internet traffic is P2P (40–73%), and BitTorrent is the most popular P2P platform accounting for 50–75% of the overall P2P traffic. Hence, focusing on "green" P2P solutions is a very sensible research direction towards an energy-friendly Internet.

Motivated by these figures and trends, in this paper we propose an energy-efficient version of BitTorrent (EE-BT) which, given its popularity, is a particularly suitable case to maximize the possible energetic impact of a novel, green P2P solution. EE-BT relies on a proxy-based architecture and is aimed at minimizing the energy consumption of user's PCs using BitTorrent for P2P file sharing. Obviously, in this paper we customize the proposed solution to the BitTorrent platform. However the ideas and concepts presented

* Corresponding author.
E-mail addresses: giuseppe.anastasi@iet.unipi.it (G. Anastasi), ilaria.giannetti@iet.unipi.it (I. Giannetti), andrea.passarella@iit.cnr.it (A. Passarella).

0140-3664/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2009.11.016

here can be easily extended to other P2P platforms as well. The proposed solution is also orthogonal to the body of research looking at legal issues such as Digital Right Management (DRM) for P2P (see, for example, Chapter 1 of [7]). Solutions for enforcing DRMs can be incorporated in our architecture. In these solutions, typically content is distributed according to a conventional P2P service, but users need to obtain a licence from the copyright owner to be able to use the content. In our architecture the same licence should be handed over by the user to the proxy. As the proxy is assumed to be trusted by the user, this is perfectly reasonable.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents our proxy-based solution for energy-efficient file sharing through BitTorrent. In Section 4 we derive some analytical formulas to quantify the energy efficiency of our solution, with respect to the legacy BitTorrent approach. In Section 5 we apply these formulas to the case of an experimental scenario and discuss the obtained results. Section 6 concludes the paper.

2. Related work

Traditional power management techniques [8] that switch the Network Interface Card (NIC) in a low-power sleep mode 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 power-management techniques compatible with persistent connectivity requirements: *adaptive link rate*, switching between *different power management levels*, and *proxy-based techniques*.

Techniques based on adaptive link rate rely on the evidence that the NIC's energy consumption strongly depends on the supported link rate. For example, the power consumption of typical Ethernet NICs increases from 1 W for 10/100 Mb/s, to 7 W for 1 Gb/s and up to 15 W for 10 Gb/s [9]. The basic idea of adaptive link rate is, thus, to adjust the link rate according to the real traffic needs. The idea is known as *Adaptive Link Rate* (ALR) [9] or *Rapid PHY Selection* (RPS) [10].

Techniques based on switching between different power management levels are targeted to NICs with different power modes, from completely sleeping to completely active (implemented as programmable FPGA technology or ASIC block technology). They switch the NIC from one mode to another, depending on the network activity, using a sleeping algorithm, such as the *Dynamic Ethernet Link Shutdown* (DELS) [11,12].

While these two techniques can provide some energy savings, they can be used only when a NIC with appropriate hardware support is available. In addition, they do not seem the best approach for our environment where downloading a file can take several hours. In this case, we believe that delegating the management of the download operations to a proxy, and shutting down the PC during the download phase, is a more effective solution. Possibly, the proxy should be running on a multi-service computer that must be always on anyway for providing other network services (e.g., DHCP, DNS etc.).

The idea of using a proxy for energy saving is not new. Proxy-based architectures have been proposed in the field of mobile computing for ensuring energy-efficient Internet access from mobile personal devices. However, in that case, the proxy architecture is intended for supporting legacy client-server applications [13]. The proxy is used as a surrogate of the mobile client on the fixed network, thus allowing the mobile device to be temporarily disconnected from the system and save energy [14].

More recently, the idea of a *Network Connectivity Proxy* (NCP) has been proposed for achieving energy efficiency in fixed as well as mobile PCs that require permanent connection to the Internet [15–17]. An NCP is an entity capable of maintaining the network presence on behalf of a sleeping PC, managing all packets destined to it. In practice, whenever receiving a packet the NCP performs one of the following actions, depending on the packet type [18]: (i) discards the packet; (ii) directly responds to the packet; (iii) re-directs the packet to another (active) computer for further processing; (iv) queues the packet for deferred processing by the PC when it wakes up; or (v) wakes up the sleeping PC and passes it the packet for appropriate processing. The NCP requires a wakeup mechanism on the sleeping PC to wake up it when necessary, e.g., a Wake On LAN (WoL) NIC [15]. The latter is a special NIC with auxiliary source power, an external wakeup signal and the capacity to recognize wakeup packets in auxiliary power. The NCP can be implemented either as part of the computer's NIC [16], or as an external entity (e.g., a USB-connected device [19], or a software module running on a router [18], switch [16] or separate computer [15,20] in the same LAN).

NCPs provide a general framework for saving energy in Internet-connected PCs during idle periods. However, they are not specifically tailored to P2P applications. Instead, our solution introduces a P2P energy-aware platform that makes possible to completely shut down the user's PC during the entire download process. Unlike NCPs, our solution does not require any specific wakeup mechanism (like the WoL NIC), which might not be available in all PCs. If available, the wakeup mechanism could be easily 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 user's PC. Instead, this can be done a later time, for example when the user re-connects.

Our proxy-based solution is also different from the Green BitTorrent proposal in [21]. There, the authors modify the legacy BitTorrent protocol to allow those peers that have already completed their download process and are not currently involved in any upload operation to put their PC in sleep mode, thus saving energy. From the viewpoint of a generic *tagged peer*, the other peers in the same swarm can be in one of the following states: *connected*, *sleeping*, and *unknown*. When the number of connected peers is less than a pre-defined threshold, the tagged peer can explicitly wakeup a sleeping peer by sending a special wakeup message to it. Green BitTorrent assumes that PCs are equipped with a WoL NIC. In addition, it introduces significant modifications in the BitTorrent protocol, although Green BitTorrent clients are compatible with legacy clients. Instead, our proposal does not require any special hardware and introduces only small modifications in the BitTorrent protocol.

The solution proposed in this paper was originally presented in [22]. In this paper we have measured in greater detail the time required to download files at the proxy, considering also maximum transfer times (instead of average values only) in case of parallel downloads. In addition, we have refined the architectural specification of the BitTorrent proxy and we have evaluated its performance gains in terms of energy consumption for different power consumptions of the involved computers. Finally, we have significantly extended the discussion on the related work.

3. Energy-efficient BitTorrent architecture

Before describing the Energy-efficient BitTorrent proposal, we provide below a brief overview of the standard BitTorrent architecture. More details can be found in [23,24].

3.1. Standard BitTorrent

BitTorrent implements an unstructured overlay network customized for file sharing [24]. In the BitTorrent terminology nodes

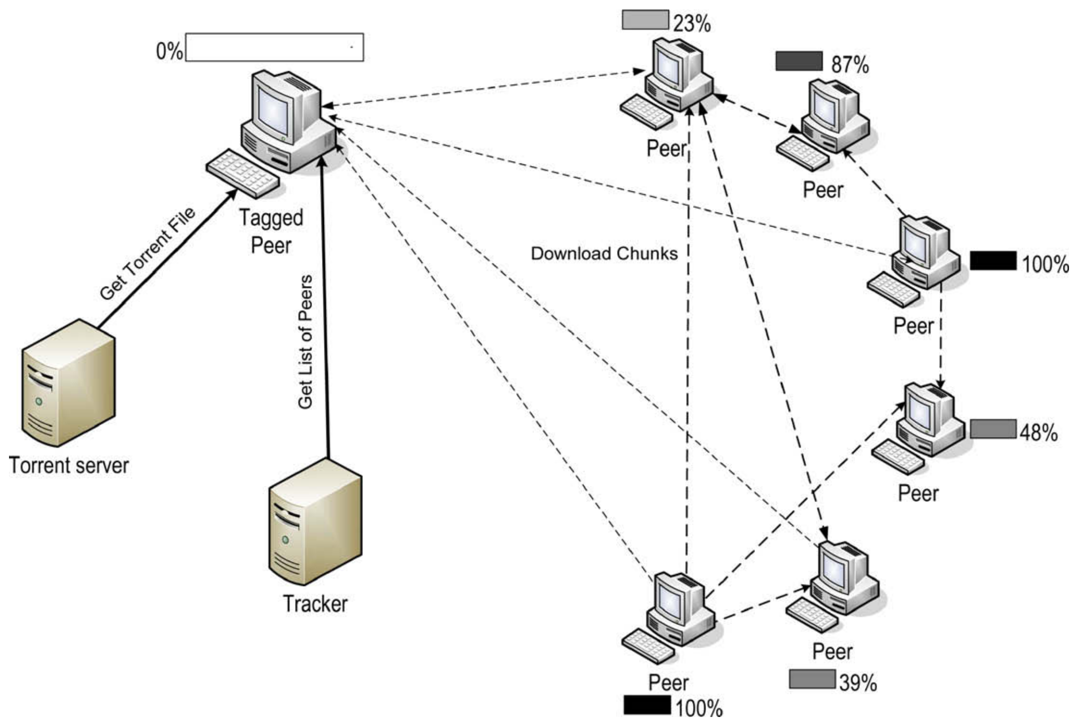

Fig. 1. File distribution process. The figure gives a snapshot of the system at the time when the tagged peer starts the download process.

of the overlay are called *peers* and the collection of peers involved in the distribution of a given file is called a *torrent* or *swarm*. The basic idea of BitTorrent is that peers both download and upload (equal-size) *chunks* of the shared files.[1] 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 [25].

As shown in Fig. 1, a tagged peer wishing to download a file from scratch needs to get a corresponding *torrent* file – hereafter referred to as torrent – from the system. Torrents are very small files, typically hosted by conventional Web servers (torrent servers), and can be found through standard Internet search engines. A torrent contains the name of the file's *tracker*. This is a node that constantly tracks which peers have chunks of the file (i.e., belong to the swarm). When a peer joins a swarm it registers with the tracker and, then, periodically informs the tracker that it is still in the swarm.

Once obtained the tracker's address, the tagged peer opens a TCP/IP connection to the tracker and receives a random list of peers to be contacted for starting the download process. At any given time the tagged peer will be in touch with a set of peers, called *neighbors*, with which it exchanges parts of the file. The neighbour set changes dynamically since, as time elapses, some peers may leave the swarm and others may join. In addition, each peer preferentially selects, for downloading chunks, those peers from which it can achieve the highest download rate (see below). Furthermore, every 30 s neighbors are selected completely at random, as a way to discover new neighbors and allow new peers in a swarm to start-up.

At a certain point in time, each peer in the swarm will have a different subset of chunks from the file. To figure out where missing chunks can be downloaded from, periodically the tagged peer asks each of its neighbors for the list of chunks they have. To decide which chunks to request first the tagged node uses the *Rarest First* policy i.e., it gives priority to those chunks that are less spread. Finally, to decide which requests from other peers to respond to, the tagged node uses the *Tit-for-Tat* (TAT) policy, i.e., it gives priority to peers from which it is downloading data at the highest rate. Specifically, for each of its neighbors the tagged node measures the downloading rate and, then, selects the four peers that are providing to it the highest bit rate.

3.2. Energy efficient BitTorrent

The legacy BitTorrent architecture is not energy efficient. BitTorrent peers have to stay connected to the overlay network during the whole download process of requested files, which, typically, may take several hours. Periodically turning off peers without modifying the BitTorrent architecture is not a viable solution for several reasons. First of all, if a peer is downloading content, powering it off does not save any energy (related to the current download), as the download itself stops when the peer turns off. Also, powering off peers that are not downloading anything (but are sharing content) is also 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 it would require central control, and is thus at odds with the BitTorrent P2P design paradigm.

In this paper we propose a proxy-based *Energy Efficient BitTorrent* (EE-BT) architecture to overcome these drawbacks. The basic idea of our architecture is illustrated in Fig. 2. We assume a standard LAN environment where a certain number of users run BitTorrent peers on their PCs. One computer in the LAN behaves as a proxy between the peers and the rest of the BitTorrent network. The proxy can either be a dedicated computer, or a machine that has to be continuously powered on for providing other network services (e.g., DHCP, Web proxy, etc.). Clearly, the latter case is preferable from an energy saving standpoint.

[1] The typical size of chunks is 256 Kbytes.

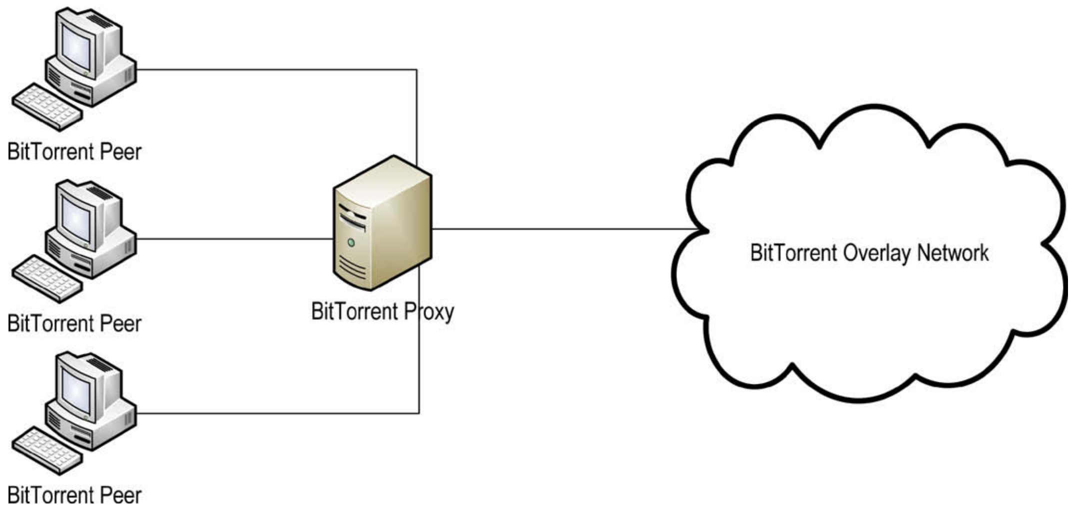

Fig. 2. High-level representation of the energy-efficient BitTorrent architecture.

Peers "behind" the BitTorrent 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 without stopping the requested downloads. Finally, the requested files are 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 overall BitTorrent network is not modified, as the proxy acts exactly as a standard BitTorrent peer. Modifications are just required at the proxy and at the user PCs 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, note that this architecture is also suitable to support mobile clients accessing the Internet, e.g., through WiFi Access Points connected to the LAN where the proxy is running, and, more in general, is a solution to enable *asynchronous BitTorrent downloads*, which is something not supported by the conventional BitTorrent architecture.

3.3. Architecture and protocols

The proposed architecture falls in the family of traditional split architectures, e.g. [26]. The architectural components between a peer and the proxy are shown in Fig. 3.

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 BitTorrent proxy and the user's PC), we adopt a simple client-server scheme, implemented by the *EE-BT Client* module at the user's PC acting as the client, and the *EE-BT Server* module at the proxy acting as the server. The EE-BT Server continuously monitors incoming requests for new downloads coming from one of the various EE-BT Clients behind it, and hands them over to the *EE-BT Daemon*. Then, the EE-BT Daemon translates these requests in download requests issued by the BT Peer running on the BitTorrent overlay network. In addition to requests for downloading new files, a BT Client can also issue commands for knowing the status of previously requested files, as well as commands to fetch the requested files from the proxy, once they have been completely downloaded. Between any successive requests, the user's PC can be turned off (or put in stand-by mode).

BitTorrent users can also upload content to the BitTorrent proxy that has to be shared on the BitTorrent overlay. This is an additional and important advantage of our architecture. The BT Peer at the proxy can share all the files that would be shared by individual peers running on the users' PCs. Therefore the BT Peer at the proxy is likely to receive more download bandwidth than any individual peer (in case no proxy is used). Thus, our proxy-based architecture is expected to achieve lower download times for all users (besides providing significant energy savings). We provide some results showing this feature in Section 5.

The proposed architecture requires very simple networking protocols. Fig. 4 shows the actions performed by the various actors during the different phases of a file download. When a user wishes to download a new file, the EE-BT Client running on the user's PC retrieves the .torrent file from a torrent server in the Internet, as in the conventional BitTorrent architecture (steps 1–2 in Fig. 4). Then, it uploads the .torrent file to the EE-BT Server on the BitTorrent proxy requesting the download of the desired file (step 3). The EE-BT Server acknowledges the received request (step 4) and hands over the .torrent file to its BT Peer – through the EE-BT Daemon – to start the download operations according to the standard BitTorrent protocol (steps 6-7). Upon receiving an acknowledgement from the EE-BT Server, notifying that the download has started, the EE-BT Client informs the user that the download request has been correctly issued and that the download process is in progress. The user can thus switch off her/his PC (step 5). As soon as the EE-BT Client on the user's PC is re-started (step 8), it checks the status of all file downloads previously requested to the EE-BT Server. For each of them, the EE-BT Client asks the EE-BT Server for a status update (steps 9–10). If the download is over, the EE-BT Client fetches the corresponding file from the proxy (steps 11–12). This transfer occurs among two computers connected through a LAN, and thus takes much less time than a typical BitTorrent download.

4. Energy efficiency analysis

To analyze the energy efficiency of our proxy-based EE-BT architecture we considered the *Absolute Energy Saving* (ΔE) and the *Relative Energy Saving* (S), defined as the absolute and relative energy savings achieved by our proxy-based architecture with respect to the legacy architecture, i.e.,

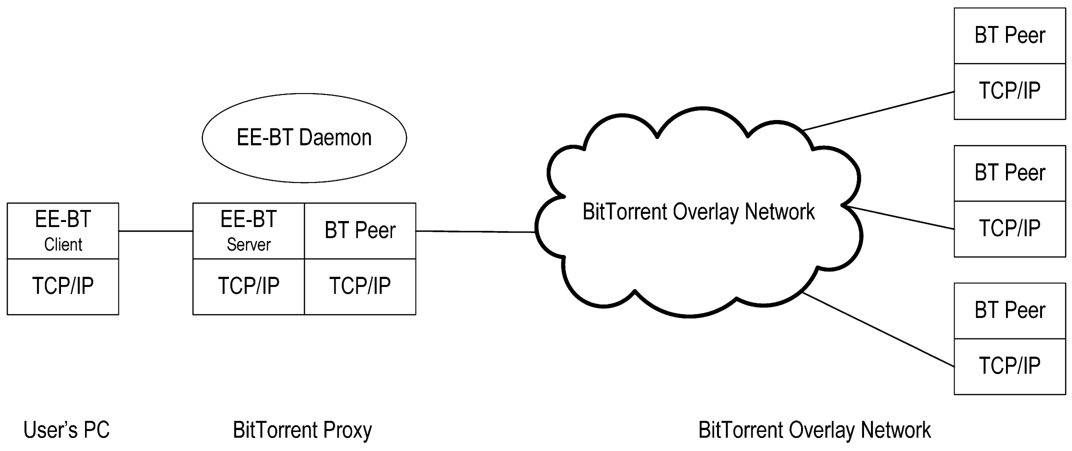

Fig. 3. Energy-efficient BitTorrent architecture.

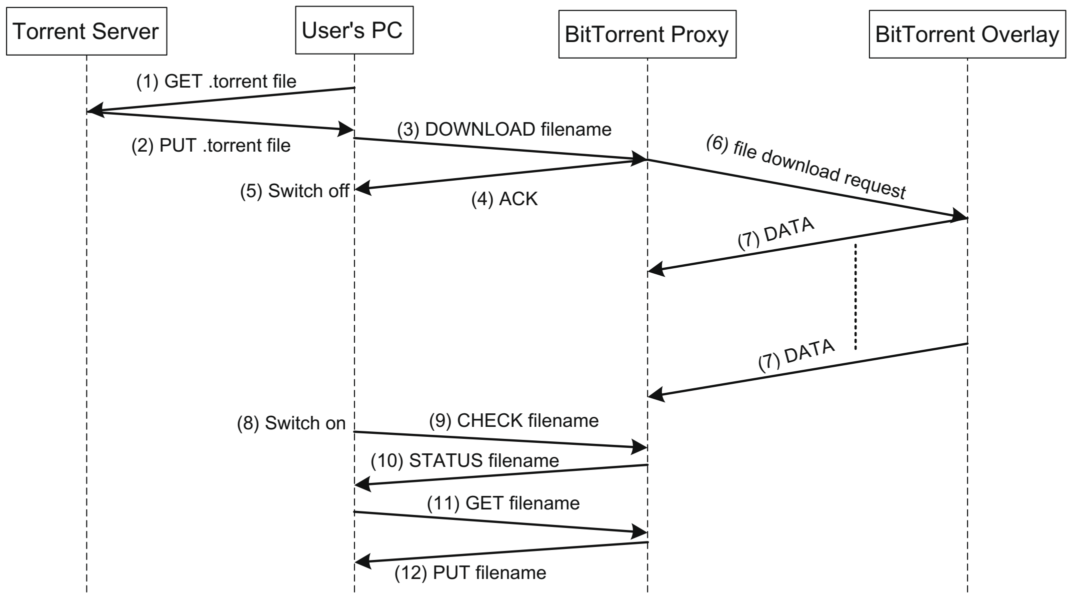

Fig. 4. Protocol actions.

$$\Delta E = E_L - E_P \tag{1}$$

$$S = 1 - \frac{E_P}{E_L} \tag{2}$$

where E_L and E_P denote the energy consumed by the user's PC to download the same file in the legacy and our proxy-based architecture, respectively.

Clearly, the energy consumed by the user's PC in both architectures is given by the total time it remains powered on, multiplied by its power consumption P_{PC}. Let us denote by t_L and t_P, the total time the user's PC must be powered on to completely download a given file, in the legacy and proxy-based architecture, respectively. As shown in Fig. 5, in the legacy architecture this time correspond to the time required by the user's PC to download the file, i.e., $t_L = d_L$. In the proxy-based architecture, in addition to the time d_P required by the proxy to download the file, we need to consider also the times taken by the user's PC for (i) delegating the file download to the proxy (t_1) and fetching the same file from the proxy, once it has been downloaded (t_2). Hence, assuming that the proxy runs on a machine that must be continuously powered on for other reasons (e.g., a multi-server machine) so that its energy consumption can be neglected, Eqs. (1) and (2) can be written as follows:

$$S' = 1 - \frac{t_P \cdot P_{PC}}{t_L \cdot P_{PC}} = 1 - \frac{t_1 + t_2}{t_L} \tag{3}$$

$$\Delta E' = (d_L - t_1 - t_2) \cdot P_{PC} \tag{4}$$

Instead, when the proxy runs on a dedicated machine, we need to consider explicitly its energy consumption. For the sake of clarity, throughout we will denote with S'' and $\Delta E''$ the relative and absolute energy saving when the proxy's energy consumption cannot be neglected.

$$S'' = 1 - \frac{t_1 + d_P + t_2}{d_L} \approx -\frac{t_1 + t_2}{d_L} \tag{5}$$

$$\Delta E'' = [d_L - (t_1 + t_2)] \cdot P_{PC} - d_P \cdot P_P \approx -(t_1 + t_2) \cdot P_{PC} \tag{6}$$

In Eq. (6), P_P denotes the power consumed by the proxy. The last passage in both (5) and (6) follows from the assumption that the BitTorrent proxy is executed on a PC similar to the user's PC and, hence, $d_L \approx d_P$ and $P_P = P_{PC}$. Note that assuming $d_L \approx d_P$ is actually a pessimistic assumption, because, as we will show in later sections,

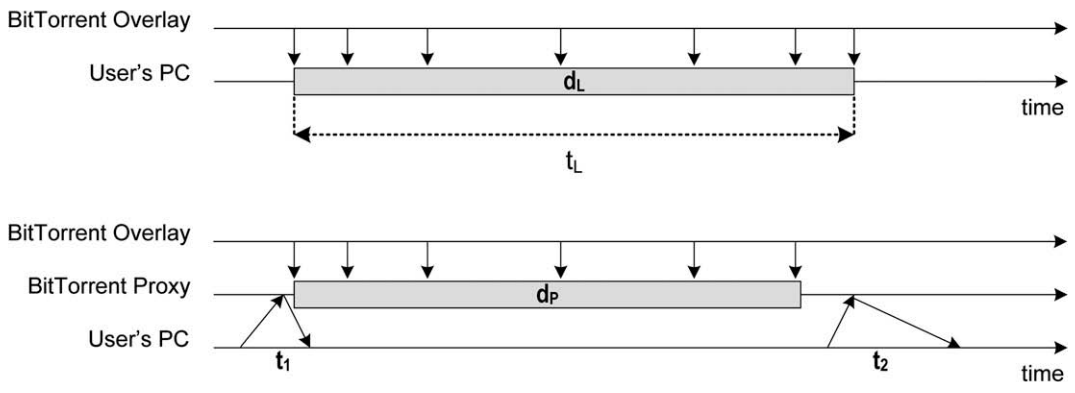

Fig. 5. File download process in the legacy (top) and proxy-based (bottom) architecture.

in general the download time when the proxy-based architecture is used is lower than when it is not. Eqs. (5) and (6) clearly show that, with a single PC, the energy consumption of the proxy-based architecture is higher than that of the legacy architecture. This is quite obvious as in the proxy-based architecture we need to consider the additional proxy's energy consumption. However, we can expect energy savings if more PCs utilize, at the same time, the BitTorrent proxy for downloading several files in parallel. Let us generalize the energy saving indices S and ΔE to the case when n different users download a file in parallel using BitTorrent on their own PC. By denoting with $S(n)$ the Relative Energy Saving when the number of PCs downloading file in parallel is n, Eqs. (3) and (5) can be generalized as follows:

$$S'(n) = 1 - \frac{\sum_{i=1}^{n} t_P(i)}{\sum_{i=1}^{n} t_L(i)} = 1 - \frac{\sum_{i=1}^{n} t_1(i) + t_2(i)}{\sum_{i=1}^{n} d_L(i)} \tag{7}$$

$$S''(n) = 1 - \frac{\sum_{i=1}^{n} t_P(i)}{\sum_{i=1}^{n} t_L(i)} = 1 - \frac{\sum_{i=1}^{n} d_P(i) + \sum_{i=1}^{n} t_1(i) + t_2(i)}{\sum_{i=1}^{n} d_L(i)} \tag{8}$$

In Eq. (8), $\sum_{i=1}^{n} d_P(i)$ denotes the total time the proxy must remain powered on for completing the download of all n files. Since the n downloads are carried out in parallel the total time corresponds to the maximum time needed to download any file, i.e., $\sum_{i=1}^{n} d_P(i) = d_P^{\max}$. Hence, Eq. (8) can be re-written as

$$S''(n) = 1 - \frac{d_P^{\max} + \sum_{i=1}^{n} t_1(i) + t_2(i)}{\sum_{i=1}^{n} d_L(i)} \tag{9}$$

Finally, following the same approach, we can also derive the Absolute Energy Saving $\Delta E(n)$ as a function of the number n of parallel file downloads, when the energy consumed by the BitTorrent proxy can, or cannot, be neglected, i.e.,

$$\Delta E'(n) = \left(\sum_{i=1}^{n} d_L(i) - \sum_{i=1}^{n} [t_1(i) + t_2(i)] \right) \cdot P_{PC} \tag{10}$$

$$\Delta E''(n) = \left(\sum_{i=1}^{n} d_L(i) - \sum_{i=1}^{n} [t_1(i) + t_2(i)] \right) \cdot P_{PC} - \sum_{i=1}^{n} d_P(i) \cdot P_P \tag{11}$$

Since $\sum_{i=1}^{n} d_P(i) = d_P^{\max}$, and assuming $P_P = P_{PC}$, Eq. (11) can be re-written as

$$\Delta E''(n) = \left(\sum_{i=1}^{n} d_L(i) - \sum_{i=1}^{n} [t_1(i) + t_2(i)] - d_P^{\max} \right) \cdot P_{PC} \tag{12}$$

5. Experimental evaluation

To evaluate the energy efficiency indices introduced in the previous section we need to derive the various times during which the user's PC(s) and proxy must remain powered on, in a real case. To this end we set up an experimental testbed and measured the delay components required for calculating both S and ΔE through the expression derived in the previous section. Below, after a brief description of the experimental testbed, we will discuss the obtained results.

5.1. Testbed description

The experimental setup was based on a set of PCs interconnected by a Gigabit Ethernet LAN which was, in turn, connected 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 PCs use Linux Ubuntu 8.04 (Hardy Heron). The BitTorrent client (i.e., the software implementing a BitTorrent peer) is a simple command-line client provided with Rasterbar libtorrent.

By exploiting the two developed systems, we performed a large set of experiments and, specifically, measured the download time (and its components) required to download the same set of files in the legacy and proxy-based architecture. More precisely, for each experiment we identified a given number, n, of files to download and we assigned one download operation to each PC. To increase the accuracy of the measurements, the same experiment was repeated several times, using always the same number n of files but changing every time the set of files. To achieve comparable statistics we selected files that are approximately of the same size and have similar popularity. Specifically, we considered file sizes in the range [3.95 GB, 4.71 GB]. For each file the initial number of *seeds* (i.e., peers that already have the whole file) was in the range [200, 800]. To have similar experimental conditions, the experiments (with and without proxy) were interleaved, so that compared results are obtained with similar congestion conditions of the Internet, and a similar number of peers.[2]

5.2. Experimental results

5.2.1. Energy savings

We start our analysis by investigating the energy savings provided by our proxy-based architecture, with respect to the legacy one.

Table 1 shows the average values of delay components experienced by each single PC and the BitTorrent proxy with different number of PCs (i.e., parallel file downloads), while Fig. 6 shows the Relative Energy Savings provided by our proxy-based solution,

[2] 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.

Table 1
Average delays experienced by each single user's PC and BitTorrent proxy for an increasing number of parallel file downloads.

Number of PCs	d_L (s)	t_1 (s)	t_2 (s)	d_P (s)	$d_P^{\max}$ (s)
1	8022.67	0.16	378.41	9288.53	9288.53
2	8927.82	0.32	756.82	5240.03	7596.92
3	6168.82	0.48	1135.23	4130.78	6084.92
4	7719.75	0.42	1246.50	3756.99	7612.89
5	9261.90	0.80	1892.05	6419.13	15012.28
6	10967.55	0.96	2270.45	8725.34	15104.89

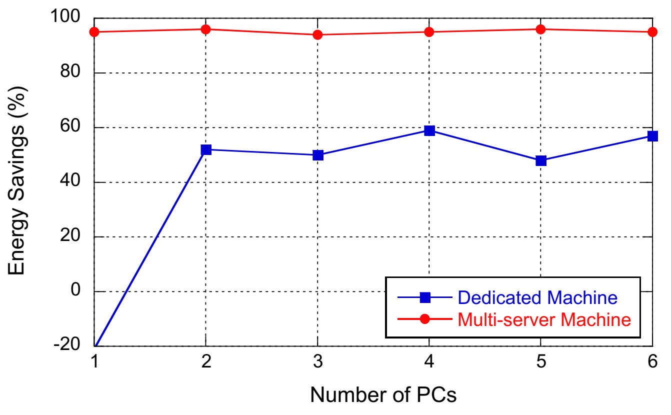

Fig. 6. Percentage energy savings vs. number of PCs.

with respect to the legacy architecture. If the BitTorrent proxy is running on a multi-server machine that must remain powered on for other purposes (and, hence, the energy consumed by the proxy can be neglected), the Relative Energy Savings $S'(n)$ do not depend on the number n of PCs and are approximately equal to 95% for each PC. Instead, when the proxy is running on a dedicated machine, so that its energy consumption must be taken into account, the Relative Energy Savings $S''(n)$ increase with the number of PCs simultaneously involved in a file download – as the proxy cost is subdivided between an increasing number of files.

These results can be easily explained by looking at Eqs. (7) and (9) . Assuming that (i) all files have approximately the same download time (i.e., $d_L(i) = d_L, \forall i$), (ii) all clients experience approximately the same delay for uploading the request to the BitTorrent proxy and downloading the file from the proxy itself (i.e., $t_1(i) = t_1,\ t_2(i) = t_2,\ \forall i$), and (iii) the interference among clients is negligible due to the extremely large bandwidth of the LAN, Eq. (7) can be written as

$$S'(n) = 1 - \frac{n \cdot (t_1 + t_2)}{n \cdot d_L} = 1 - \frac{t_1 + t_2}{d_L} = S'(1) \tag{13}$$

This means that, if the proxy runs on a multi-server machine, the Relative Energy Saving achieved by each single PC does not depend on the number of files to download in parallel. This also suggests that the Absolute Energy Saving is (approximately) linearly increasing with the number of PCs. We show this in more detail in the following. First, we compute $\Delta E'(n)$ and $\Delta E''(n)$ by introducing the delay components measured in our experiments into Eqs. (10) and (12), respectively. As far as the power consumption of PCs and BitTorrent proxy, we need to emphasize that it strongly depends on the computer type (e.g., desktop or laptop). In addition, for the same machine, it varies over time depending on the operational mode. In [19] it is shown that typical power consumption values for a PC in normal idle state are in the range 70–100 W for desktop machines and 15–30 W for laptops. Therefore, in calculating the absolute energy savings provided by our proxy-based architecture we considered two different values of power consumption, i.e., 30 and 100 W. The obtained results are plotted in Fig. 7. They clearly show that, in both cases, the Absolute Energy Savings increase almost linearly with the number of PCs (i.e., number of parallel file downloads). For both values of power consumption taken into consideration, the difference between the two corresponding curves is only due to the proxy's energy consumption.

Again, it is possible to provide an analytical explanation for the behaviors observed in Fig. 7. By following the same arguments used above, it is easy to show that $\Delta E'(n)$ increases almost linearly with the number of files:

$$\Delta E'(n) \approx n \cdot [d_L - (t_1 + t_2)] \cdot P_{PC} = n \cdot \Delta E'(1) \tag{14}$$

On the other hand, when we include the energy consumed by the proxy in the total energy consumption, we obtain (from Eq. (12))

$$\Delta E''(n) \approx \left\{ n \cdot [d_L - (t_1 + t_2)] - d_P^{\max} \right\} \cdot P_{PC} \tag{15}$$

Assuming that $d_L \approx d_P^{\max}$ Eq. (15) can be approximated as:

$$\Delta E''(n) \approx [(n-1) \cdot d_L - n \cdot (t_1 + t_2)] \cdot P_{PC} \xrightarrow{n} \Delta E'(n) \tag{16}$$

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

$$\begin{aligned} S''(n) &\approx 1 - \frac{n \cdot (t_1 + t_2) + d_P^{\max}}{n \cdot d_L} \\ &\approx 1 - \frac{(t_1 + t_2)}{d_L} - \frac{1}{n} \xrightarrow{n} 1 - \frac{(t_1 + t_2)}{d_L} = S'(n) \end{aligned} \tag{17}$$

Eqs. (16) and (17) show that, for large n values, the BitTorrent proxy's energy consumption can be neglected. This trend does not emerges very clearly from Figs. 6 and 7 because the considered number of parallel file downloads (i.e., n) is not so large for practical reasons. In addition, even replicating each single experiment several times, the obtained results are characterized by a high variability due to a number of reasons. First, the network conditions change over time. In addition, the considered file have not exactly the same size. Finally, the time to completely download a file is largely dependent on the number and positions of peers having that file.

5.2.2. File download time

The results presented above clearly show the effectiveness of the proxy-based architecture from the energy efficiency standpoint. We will now investigate the impact of the BitTorrent proxy on the Quality of Service (QoS) perceived by the user, i.e., on the file download time. We will show that the proxy-based architecture does not introduce any QoS degradation. Instead, with respect to the legacy approach, it reduces significantly the average file download time.

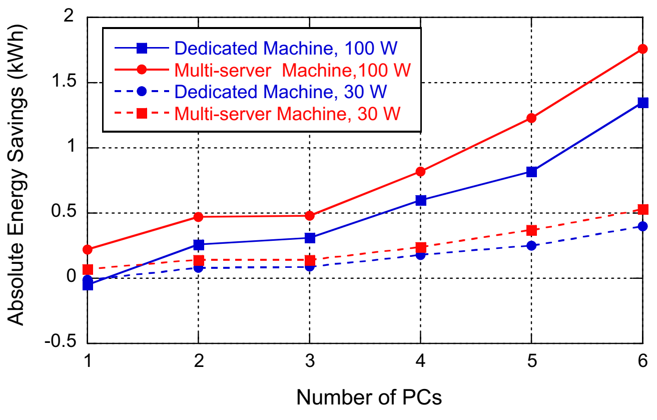

Fig. 7. Absolute energy savings vs. number of PCs.

In the previous analysis we have assumed that the time to download a file is not significantly affected by the proxy's presence. To analyze this aspect we performed a set of experiments and measured the time needed to download *n* files in parallel, with the legacy and proxy-based architectures, respectively. The results of this analysis are summarized in Fig. 8, where we plot the average download time of a single file, for an increasing number of parallel file downloads (each column in Fig. 8 is the average calculated on all replicas). The results shown in Fig. 8 clearly indicate that the BitTorrent proxy does not introduce any degradation in the QoS experienced by user. Instead, on average, the time for downloading a file *reduces* when using the proxy-based architecture. This can be explained by considering that the peer running on the BitTorrent proxy shares more files on the overlay with respect to any single peer in the legacy architecture, and thus gets higher download bandwidth. To quantify the average gain we can achieve when using our proxy-based architecture, we computed the average time to download a file in all the experiments we performed. The average download times were 6541s with the BitTorrent proxy and 8439s with legacy BitTorrent. Thus, the BitTorrent proxy reduces the average download time by approximately 22%.

We wish to conclude our analysis on the file download time by emphasizing an interesting aspect that is tightly coupled with the BitTorrent behavior. Specifically, we analyze how the availability on the proxy of a single (popular) file to upload can highly reduce the download time of all files the proxy is downloading. This effect is well exemplified by the results in Fig. 9 where we show how the average delay experienced by the BitTorrent proxy to download the same file is affected by the presence of a popular file on the proxy itself. We performed two set of experiments where the proxy is downloading, in parallel, 3 and 4 different files, respectively. For each set we considered the cases when a popular file *is* or *is not* available on the proxy. As it clearly appears from the results in Fig. 9, a single popular file can further reduce – with respect to the gain already achieved with the proxy – the download time of all the other files by 25–30%. In addition to energy efficiency, this provides a strong motivation for exploiting our proxy-based architecture: a single popular file shared on the proxy provides a high benefit to everyone. This suggests to adopt policies consisting in selecting popular files available on client to be uploaded on and shared by the proxy, as a way to optimize the overall download performance of the system.

6. Conclusions

In this paper we have proposed a BitTorrent Proxy for energy-efficient P2P file sharing over the Internet. It is a mechanism for saving energy in an environment where users leave their PCs powered on for sharing file through BitTorrent. The problem is due to permanent connectivity requirements which make the use of traditional power management techniques unsuitable. To overcome this problem we have proposed a proxy-based BitTorrent architecture where file downloads are delegated to a proxy. Our goal was saving energy at the user's PCs without introducing any significant degradation of the QoS, in particular without increasing the file download time.

We have evaluated our solution in a realistic testbed, measuring the file download time with the legacy and proxy-based architectures, respectively. Our experimental results have shown that the proxy-based architecture can save up to 95% of the energy consumed by each PC when using the legacy solution. This shows the effectiveness of our approach from the energy efficiency standpoint. In addition, our results have shown that using the BitTorrent Proxy does not introduce any degradation of the QoS. Rather, the average time to download a file *reduces* by approximately 22% when using the proxy-based architecture since the number of files shared with the overlay network by the proxy is greater then the number of files shared by any single peer. Finally, we have also observed that the presence on the proxy of a very popular file to upload – while the proxy is downloading files for its clients – further reduces the average download time by 25–30%, showing additional benefits of using our proxy-based architecture.

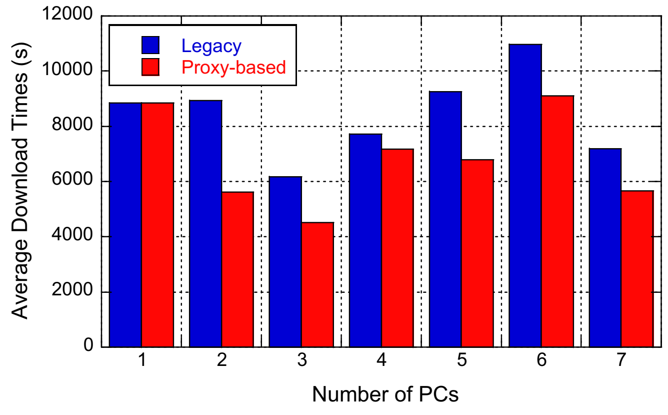

Fig. 8. Average download time experienced by each single file in the legacy and proxy-based architecture.

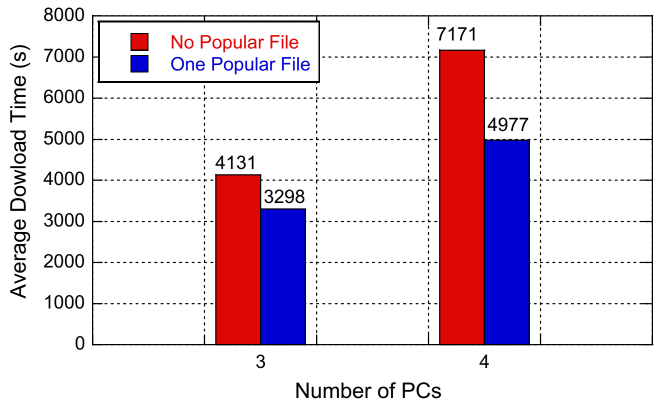

Fig. 9. Impact on the proxy's download time of a popular file distributed in parallel to 3 (a) and 4 (b) peers.

References

[1] K. Christensen, A.D. George, Increasing the Energy Efficiency of the Internet with a Focus on Edge Devices, The Energy Efficient Internet Project, University of South Florida and University of Florida, Florida, 2005–2008.
[2] S. Ruth, Green IT – more than a three percent solution, IEEE Internet Computing Magazine 13 (4) (2009). July–August.
[3] G. Goth, The net's going green, IEEE Computer (1) (2008) 7–9.
[4] C. Gunaratne, K. Christensen, S. Suen, B. Nordman, Reducing the energy consumption of ethernet with an adaptive link rate, IEEE Transactions on Computers 57 (4) (2008). April.
[5] National Energy Foundation (NEF), S. Karayi, The PC energy report, 1E, London, 2007, <http://www.1e.com/energycampaign/downloads/1E_reportFINAL.pdf>.
[6] H. Schulze, K. Mochalski, 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, September 2007.
[7] J. Buford, H. Yu, E.K. Lua, P2P Networking and Applications, Morgan Kaufmann Publishers Inc., 2009.
[8] G. Anastasi, M. Conti, A. Passarella, Power management in mobile and pervasive computing systems, in: Azzedine Boukerche (Ed.), Chapter 24 in Algorithms and Protocols for Wireless and Mobile 12 Networks, CRC Computer and Information Science Publisher, 2005. October.

[9] C. Gunaratne, K. Christensen, Ethernet adaptive link rate: system design and performance evaluation, in: Proceedings IEEE Conference on Local Computer Networks, November 2006, pp. 28–35.

[10] K. Christensen, F. Blanquicet, An initial performance evaluation of rapid PHY selection (RPS) for energy efficient Ethernet, in: Proceedings IEEE Conference on Local Computer Networks, October 2007, pp. 223–225.

[11] M. Gupta, S. Grover, S. Singh, A feasibility study for power management in LAN switches, IEEE ICNP 2004, Berlin, Germany, October 2004.

[12] S. Singh, M. Gupta, Using low-power modes for energy conservation in Ethernet LANs, in: INFOCOM 2007 26th IEEE International Conference on Computer Communications IEEE, Portland State University, Portland, May 2007.

[13] G. Anastasi, M. Conti, E. Gregori, A. Passarella, Performance comparison of power saving strategies for mobile Web access, Performance Evaluation 53 (3–4) (2003) 273–294. August.

[14] E. Pitoura, G. Samaras, Data Management for Mobile Computing, Kluwer Academic Publishers, 1998.

[15] K. Christensen, F. Gulledge, Enabling power management for network-attached computers, International Journal of Network Management 8 (2) (1998) 1099–1190).

[16] C. Gunaratne, K. Christensen, B. Nordman, Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP, connections, and scaling of link speed, International Journal of Network Management 15 (5) (2005) 297–310. September/October.

[17] K. Christensen, B. Nordman, Improving the Energy Efficiency of Ethernet – Connected Devices: A Proposal for Proxying, Ethernet Alliance White Paper, September 2007, <http://efficientnetworks.lbl.gov/enet-pubs.html>.

[18] M. Jimeno, K. Christensen, B. Nordman, A network connection proxy to enable hosts to sleep and save energy, in: IEEE International Performance Computing and Communications Conference, December 2008, pp. 101–110.

[19] Y. Agarwal, S. Hodges, J. Scott, R. Chandra, P. Bahl, R. Gupta, Somniloquy: augmenting network interfaces to reduce PC energy usage, in: Proceedings USENIX Symposium on Networked System Design and Implementation (NSDI, 2009), Boston, MA, USA, April 2009.

[20] S. Nedevschi, J. Chandrashekar, B. Nordman, S. Ratnasamy, N. Taft, Skilled in the art of being idle: reducing energy waste in networked systems, in: Proceedings USENIX Symposium on Networked System Design and Implementation (NSDI, 2009), Boston, MA, USA, April 22–24, 2009.

[21] J. Blackburn, K. Christensen, A simulation study of a new Green BitTorrent, in: Proceedings First International Workshop on Green Communications (GreenComm 2009), Dresden, Germany, June 2009.

[22] G. Anastasi, M. Conti, I. Giannetti, A. Passarella, Design and evaluation of a BitTorrent proxy for energy saving, in: Proceedings IEEE Symposium on Computers and Communications (ISCC 2009), Sousse, Tunisia, July 2009.

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

[24] J. Kurose, K. Ross, Peer-to-peer Applications, Computer Networking. A Top-Down Approach, fourth ed., Addison Wesley, 2007.

[25] D. Towsley, The Internet is flat: a brief history of networking over the next ten years, ACM PODC 2008, 2008.

[26] 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 46 (1) (2003). January.