ARTICLE TYPE

A Sleep-and-Wake technique for reducing energy consumption in BitTorrent networks

Fabrizio Marozzo* | **Domenico Talia** | **Paolo Trunfio**

[1]DIMES, University of Calabria, Rende, Italy

Correspondence
*Fabrizio Marozzo Email: fmarozzo@dimes.unical.it

Present Address
DIMES, University of Calabria, Via P.Bucci 41C, 87036 Rende, Italy

Summary

File sharing is one of the leading Internet applications of P2P technology. Given the high number of computer nodes involved in peer-to-peer networks, reducing their aggregate energy consumption is an important challenge to be faced. In this paper, we show how the sleep-and-wake energy saving approach can be exploited to reduce energy consumption in BitTorrent, one of the most popular file sharing peer-to-peer networks. We describe BitTorrentSW, a sleep-and-wake approach for BitTorrent networks that allows seeders (i.e., peers that hold complete files) to cyclically switch between wake and sleep mode to save energy while ensuring good file sharing performance. The decision to switch to sleep mode is taken independently by each seeder based on local information about the composition of the peer-to-peer network. BitTorrentSW has been evaluated through PeerSim using real BitTorrent traces. The simulation results show that, in all the configurations under analysis, the percentage of energy saved by BitTorrentSW is much higher than the percentage of increase in download time. For instance, in a network with 50% of seeders, about 20% of energy is saved using BitTorrentSW, with an increase of only 7% of the average time needed to complete a file download compared to a standard BitTorrent network in which all seeders are always powered on.

KEYWORDS:
Peer-to-peer, BitTorrent, File sharing, Energy efficiency, Energy saving, Performance analysis

1 | INTRODUCTION

As file sharing peer-to-peer networks involve a very large number of hosts, their aggregate energy consumption is an important problem to be addressed, given the economic and environmental impact of energy production and use. Several researchers have proposed solutions for saving energy of peer-to-peer networks such as message reduction, location-based mechanisms, proxies, optimizing task allocation or overlay structure, and the "sleep-and-wake" strategy [1]. The sleep-and-wake is one of the most important approaches, considering that the energy consumption of a peer-to-peer network can be significantly reduced if peers periodically switch from "wake" mode (high-power) to "sleep" mode (low-power). In fact, a main cause of energy waste in a peer-to-peer network are peers powered on even when they are not doing upload/download activities.

In this paper, we show how the sleep-and-wake energy saving approach can be exploited to reduce energy consumption in BitTorrent, the dominant file sharing protocol and one of the applications that generate more traffic on Internet, with 2.46% of downstream and 27.58%

of upstream traffic [2]. To reach this goal we developed *BitTorrentSW*, a sleep-and-wake technique for BitTorrent networks that allows peers to cyclically switch between wake and sleep mode for saving energy while ensuring good file sharing performance. In BitTorrent, peers can be *seeders* or *leechers*: the former hold complete files and share them; the latter are downloading the parts they need to complete the file, and share the parts they already have. In BitTorrentSW only the seeders can switch to sleep mode. The decision is taken autonomously by each seeder according to the local information it owns about network composition: if the percentage of seeders in the network exceeds a certain threshold, the seeder switches to sleep mode for a period of time.

A few other sleep-and-wake techniques have been proposed to improve the energy efficiency of BitTorrent networks [3 4 5], but they differ from BitTorrentSW in three main aspects: i) all of them turn off any peer that is not doing upload/download activities, while BitTorrentSW puts in sleep mode only the seeders that are not doing upload activities; ii) some of them modify the BitTorrent protocol by introducing new messages, while BitTorrentSW does not modify the BitTorrent protocol with the introduction of new messages; iii) some of them assume that hosts are equipped with a Wake-on-LAN connector that allows a peer to be re-awakened with a message, while BitTorrentSW can be run on any hardware as wake up is scheduled locally. It is worth noticing that the BitTorrentSW scheme is backward compatible, as a mix of BitTorrentSW and standard BitTorrent peers can co-exist in the same network. In fact, as specified above, BitTorrentSW does not modify the BitTorrent protocol with the introduction of new messages, which means that standard peers will not receive any unknown message to process. In addition, when a peer switches to sleep mode, it is seen by all the other peers as if it has left the network, and thus no additional actions must be carried out.

BitTorrentSW has been evaluated through PeerSim [6] using parameters extracted from real BitTorrent traces. The simulation results show that, in all the configurations under analysis, the percentage of energy saved by BitTorrentSW is much higher than the percentage of increase in download time. For instance, in a network with 50% of seeders, about 20% of energy is saved using BitTorrentSW, with an increase of 7% of the average time needed to complete a file download compared to a standard BitTorrent network in which all seeders are always powered on. The results are further improved when the percentage of seeders in the network increases. For example, with 60% of seeders, we registered 28% of energy saving with only 4% of increase in download time. To summarize, all the results demonstrate the effectiveness of BitTorrentSW in reducing energy consumption with a negligible impact on the average download time. This paper extends an earlier version of the work [7], by describing in depth the sleep-and-wake algorithm implemented by BitTorrentSW, and providing a wider set of simulation results based on real BitTorrent traces described in [8], [9], [10] and [11].

The remainder of the paper is structured as follows. Section 2 discusses related work. Section 3 presents the system model. Section 4 describes the BitTorrentSW algorithm. Section 5 presents an evaluation of the energy-saving algorithm using PeerSim. Finally, Section 6 concludes the paper.

2 | RELATED WORK

Several solutions have been proposed for improving the energy-efficient of large-scale computing systems [12,13,14]. In this section, we focus on the main solutions proposed to enhance energy efficiency of peer-to-peer systems, some of which specifically designed for BitTorrent networks.

Malatras et al. [1] classified existing solutions for improving the energy-efficient of peer-to-peer networks in six categories:

1. The *proxying approach* allows peers to delegate some of their activities to proxies, such as file downloading. Using proxies, peer-to-peer hosts do not need to stay constantly on-line, this way reducing the overall energy consumption of the peer-to-peer networks. Examples of proxy-based approaches are the system by Purushothaman et al. [15] for Gnutella networks [16], and the system proposed by Anastasi et al. [17] for reducing the energy consumption of hosts running the BitTorrent application.

2. *Task allocation optimization* reduces the overall energy consumption in peer-to-peer networks by carefully scheduling the allocation of tasks to peers, i.e., deciding on which peer will satisfy the request of another peer. One example is the work by Enokido et al. [18], who proposed a model for peer-to-peer data transfers in which computation time and power consumption are minimized by optimizing the allocation of file requests.

3. *Message reduction* aims at reducing the number of messages exchanged through the peer-to-peer network for lowering processing and transmission times, thus reducing energy consumption. One example of energy-saving peer-to-peer system based on this approach is the work by Kelenyi and Nurminen [19], who adopted a selective message dropping mechanism for reducing the number of messages exchanged in a Kademlia network [20].

4. *Overlay structure optimization* improves the energy efficiency of a peer-to-peer network by controlling its topology during construction, and by introducing new layers to the overlay. An example of the first type is the work by Leung and Kwok [21], where topology control is used for improving the energy efficiency of wireless file sharing peer-to-peer networks. An example of the second type is the double-layered system by Han et al. [22].

5. The *location-based approach* exploits positioning information about nodes for making peer-to-peer overlays more closely matching the underlying physical connections with the goal of reducing multi-hop transmissions, and consequently the overall energy consumption. This approach is particularly effective in mobile peer-to-peer networks, as proven by the research works proposed by Joseph et. al [23], Park and Valduriez [24], and Tung and Lin [25].

6. The *sleep-and-wake approach* reduces the overall energy consumption of a peer-to-peer network by allowing peers cyclically switch between normal and sleep state. The critical point of this approach is deciding when peers should be in normal or sleep state, so as to avoid excessive degradation of system performance. Several systems fall in this category, including the ones by Andrew et al. [26], Sucevic et al. [27], Corigliano et al. [28] and Trunfio [29].

In the following we briefly compare BitTorrentSW with the main sleep-and-wake techniques used to improve the energy efficiency of BitTorrent networks. Blackburn and Christensen [3] proposed *green BitTorrent*, a BitTorrent extension for optimizing energy consumption through a sleep-and-wake approach. Such approach allows peers to switch to sleep mode when they are not doing any operations (i.e., downloading/uploading chunks), but keeping them active members of the network. The basic idea is similar to that used in our approach with these main differences: i) green BitTorrent turns off any peer (seeders and leechers) that is not doing upload/download activities; ii) it modifies the BitTorrent protocol by introducing new messages related to the awakening of the peers; iii) it assumes that hosts are equipped with a Wake-on-LAN (WoL) connector which allows the peer to be re-awakened with a message. Our approach switches to sleep mode only the seeders that are not doing uploading activities, it does not modify the BitTorrent protocol with the introduction of new messages, and can be run on any hardware.

Forshaw and Thomas [4] proposed a solution for saving energy in BitTorrent networks. Basically they consider a seed pool, that is a group of seeders waiting to share content to leechers of the network by trying to guarantee a satisfactory level of performance. The number of seeders is elastic for adapting to the requests of real-time services. The main difference with our solution is that Forshaw and Thomas have modified the BitTorrent protocol to allocate the upload bandwidth based on the combination of download rates and energy efficiency. More specifically, a seeder will send the file chunks to those peers that have a higher power consumption so that they finish the download as quickly as possible for reducing the energy consumption of the entire network. Unlike our algorithm, therefore, the implementation of Forshaw and Thomas is not compatible with the legacy algorithm of BitTorrent as the conditions for choosing the leechers to be served are different from the original ones.

Lee et al. [5] describes a methodology for improving the energy efficiency of a BitTorrent network through the definition of new states for the peers involved in. The communication of information related to these new states is achieved through specific hibernation and awakening messages. In addition, the peers in standby mode are awakened via the WoL technology, re-establishing the TCP connections. A main difference with our approach is that the solution by Lee et al. uses custom messages for communicating changes of status (e.g., a node that goes in standby communicates to the tracker its status change). In our approach, it is used a pure BitTorrent protocol that does not create new types of messages to communicate with the tracker. In addition, in Lee's algorithm the leechers are also put in standby mode, as opposed to what happens in our approach where only seeders can decide autonomously, based on local information, whether to go into energy saving mode or not.

3 | SYSTEM MODEL AND DESIGN PRINCIPLES

In BitTorrent, a file F is described by a *torrent*, i.e., a descriptor containing file metadata (name, size, hash values for verifying its integrity) and the network locations of one or more trackers (as defined below). Each file is split into small pieces (with a size of 256 kB or 512 kB), called *chunks*.

The BitTorrent architecture includes three basic roles:

- A *tracker* keeps track of where file chunks reside on peers, and which peers are available. Trackers are not directly involved in file transfers and do not have a copy of the file.

- A *seeder* is a peer that holds the complete file (i.e., all the chunks) and shares it.
- A *leecher* is a peer that does not hold all the file chunks; it is downloading the remaining chunks and can share the chunks it owns.

Figure 1 shows an example of BitTorrent network that share a file F. In this example, F is divided into six chunks (F_1...F_6). A set of *seeders*, $S_1 \ldots S_5$ own a copy of F, and a set of *leechers*, $L_1 \ldots L_8$ are trying to complete the download of F. A tracker T keeps track of which chunks are owned by the peers (leechers and seeders). For example, L_1 owns one chunk, L_2 owns two chunks, L_3 does not own any chunk, and so on.

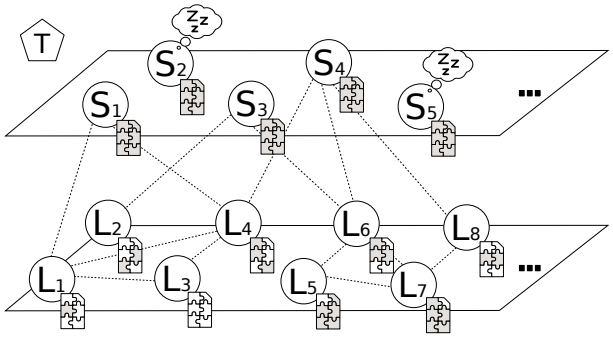


FIGURE 1 An example of BitTorrent network composed by a tracker T, a set of seeders $(S_1 \ldots S_5)$ and a set of leechers $(L_1 \ldots L_8)$ that share a file F.

The *BitTorrentSW* approach is based on the idea of keeping in sleep mode seeders that are not performing upload activities (for example, seeders S_2 and S_5 in Figure 1). To avoid the need for centralized coordination, the decision of switching to sleep mode is taken autonomously by each seeder according to the local information it owns about network composition. Specifically, each peer maintains a *peer set* (also called *neighbor set*), that is a data structure containing the neighbors of a generic peer, with which the peer can establish a connection and perform download and upload operations. The peer set is kept constantly updated in BitTorrent to provide each node with a wide choice of possible connections to be established. In BitTorrentSW, a seeder can switch to sleep mode for a period of time if the percentage of seeders in its peer set exceeds a certain threshold. Details on the sleep-and-wake algorithm will be provided in the next section.

4 | SLEEP-AND-WAKE ALGORITHM

Following the sleep-and-wake approach, it is assumed that each seeder periodically switches from normal (or wake) mode to sleep mode, and vice versa. When a seeder is in *normal* mode, it is *available* for download requests and works at normal power level (p_{high}). Conversely, a seeder in *sleep* mode is *unavailable*, but it works at reduced power level (p_{low}), thus consuming a limited amount of energy. Figure 2 illustrates the relationship between availability status of a seeder and its power mode.

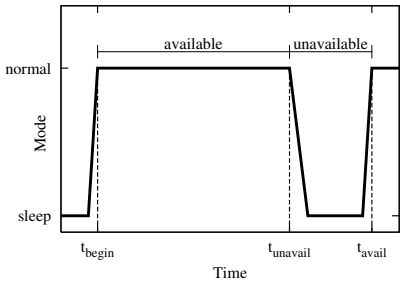


FIGURE 2 Relationship between availability status and power mode of a seeder.

Figure 3 shows the UML state diagram that describes the behavior of a peer in a BitTorrentSW network. A newly created peer that does not own the file becomes a *Leecher*. When it has downloaded the whole file (or if it is a file owner) it becomes a Seeder. The Seeder state is a macro-state composed by two sub-states: *Seeder.Active*, which is a seeder running in normal power; *Seeder.Sleeping*, which is a seeder in sleep mode. *Seeder.Active* is in turn a macro-state composed by two sub-states: *Seeder.Active.NotUploading*, which represents an active seeder not performing any upload operation; *Seeder.Active.Uploading*, for a seeder that is performing a least one upload operation. If a seeder is not performing any upload operation, it can change its state from *Seeder.Active* to *Seeder.Sleeping*. The decision is taken autonomously by each seeder according to the local information it has about the network: if the percentage of seeders present in its peer set (*currentPercSeeders*) exceeds a certain threshold (*desiredPerc*), the seeder goes sleeping for a certain amount of time (*sleepingTime*). After *sleepingTime*, the seeder will return to be active (*Seeder.Active*).

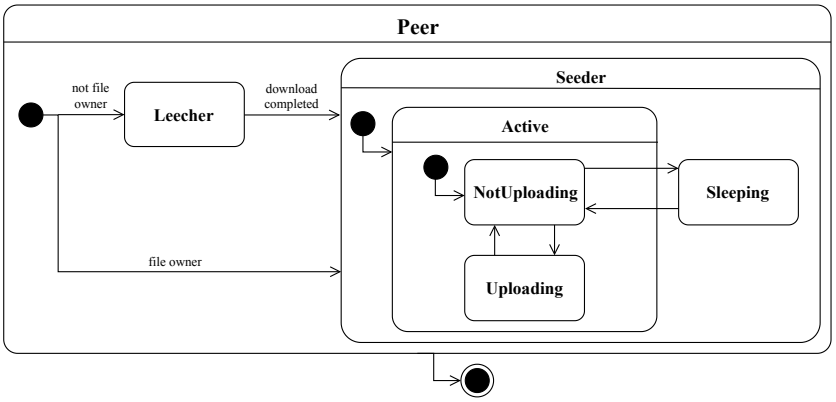


FIGURE 3 State diagram of a generic peer in BitTorrentSW.

Algorithm 1 BitTorrentSW peer set maintenance algorithm.

```
 1: procedure P_i.peer_set_maintenance()
 2:     peerSet = tracker.getInitialPeerSet()
 3:     while true do
 4:         for each peer in peerSet do
 5:             if peer.isNotActive then
 6:                 peerSet.remove(peer)
 7:             end if
 8:         end for
 9:         if peerSet.size()<minPeerSet then
10:             peerSet.addAll(tracker.getNewPeers())
11:         end if
12:         power_management()
13:         wait for checkTime seconds
14:     end while
15: end procedure
```

Algorithm 1 shows the main operations done by a peer to keep updated its peer set. At first, a peer gets the initial list of neighbors from the tracker (line 2). Periodically, the peer checks the connections with its neighbors and eliminates those that are no longer active (lines 5-7). If the number of neighbors drops below a certain threshold (*minPeerSet*), the peer contacts the tracker to obtain some additional peers to add to its peer set (lines 9-11). These operations (lines 4-12) are re-executed every *checkTime* seconds (line 13). Compared to the original BitTorrent algorithm, a *power_management* procedure implemented the sleep-and-wake strategy is executed at line 12.

Algorithm 2 describes the *power_management* procedure. If the peer is an active seeder not performing any upload operation (line 2), it calculates the current percentage of seeders according to local information (line 3). This is done by calculating the percentage of seeders

present in its peer set. Then, the seeder calculates the difference between *currentPercSeeders* and the desired percentage of seeders *desiredPerc* (line 4). If such difference is greater than zero, it means that there is an excess of seeders in the network (line 5). Thus, the status of the seeder is changed to *Seeder.Sleeping* (line 6) and the seeder switches to sleep mode for a given *sleepingTime* (line 7). After the sleeping time, the seeder wakes up and returns to be active (line 8). Being the power management procedure included in the peer set maintenance algorithm, it is invoked every *checktime* seconds.

Algorithm 2 BitTorrentSW power management algorithm.

```
1:  procedure P_i.power_management()
2:      if status = Seeder.Active.NotUploading then
3:          currentPercSeeders := getPercSeeders(peerSet)
4:          difference = currentPercSeeders - desiredPerc
5:          if difference > 0 then
6:              status := Seeder.Sleeping
7:              switch to sleep mode for sleepingTime seconds
8:              status = Seeder.Active
9:          end if
10:     end if
11: end procedure
```

5 | PERFORMANCE EVALUATION

The goal of this section is to evaluate the amount of energy saved by BitTorrentSW and the delay it causes to file download. To this end, two main performance parameters have been measured over a period of observation: the *total energy consumed by the network*, E_{tot}, and the *average delay to complete the download of a file*, D_{avg}. To compare the performance of BitTorrentSW compared to the standard version of BitTorrent, the values of E_{tot} and D_{avg} will be measured in two different cases: i) seeders can switch to sleep mode by executing the BitTorrentSW; ii) seeders execute the standard BitTorrent, thus remaining always in normal mode.

5.1 | Experimental methodology

An implementation of the BitTorrent protocol for PeerSim has been used to carry out the performance evaluation. The simulator works in three phases:

1. A BitTorrent network composed of N_{peers} peers is executed to share a file F; a percentage $P_{seeders}$ of peers is selected to act as seeders, while the remaining percentage of these peers is selected to play the role of leechers ($P_{leechers}$).
2. Each seeder owns a complete copy of F, while each leecher receives a random number of file chunks. Given the size of F, F_{size}, the number of file chunks is F_{size}/C_{size}, where $C_{size} = 256$kB is the standard chunk size in BitTorrent.
3. The simulation begins, with seeders performing server-side activities (file uploads) and leechers running both client- and server-side activities (file downloads and uploads).

The simulator assumes that seeders are not used by other applications, and therefore they can switch to sleep mode taking into account only their upload activities on F. Each simulation terminates when the clock reaches a value T_{sim}, which represents the simulation length. At the end, the performance parameters E_{tot} and D_{avg} are calculated by the following equations:

$$E_{tot} = \sum_{t=1}^{T_{sim}} \sum_{i=1}^{N_{peers}} p_i(t) \cdot \Delta t \quad (1)$$

where Δt is the time resolution of the simulator (10 seconds), and $p_i(t)$ is the power consumed by the i-th host at time t, which is equal to p_{low} if the host is in sleep mode at time t, p_{high} otherwise;

$$D_{avg} = \frac{1}{N_{leechers}} \sum_{i=1}^{N_{leechers}} t_{down}(i) - t_{start}(i) \quad (2)$$

where $N_{leechers}$ is the number of leechers, $t_{start}(i)$ is the instant of time when the i-th leecher started to download file F, and $t_{down}(i)$ is the instant of time when the download is complete;

5.2 | Simulation parameters

Table 1 reports the parameters used for the simulations. The most relevant parameters are extracted from real BitTorrent traces described in[8],[9],[10] and[11].

Parameter	*Description*	*Values*
N_{peers}	Number of nodes in the network	1000 - 4000
$P_{seeders}$	Percentage of seeders available in the network	40% - 60%
F_{size}	Size of the shared file	256MB - 1024 MB
T_{sim}	Simulation length	24 hours
p_{high}	Power consumed by a node in normal mode	150 W
p_{low}	Power consumed by a node in sleep mode	5 W
$T_{sleep\_to\_normal}$	Amount of time to switch from sleep to normal mode	4 s
$T_{normal\_to\_sleep}$	Amount of time to switch from normal to sleep mode	9 s
T_{ckeck}	Check time	120 s
U_{rate}	Upload rate	640 - 4096 Kb/s
T_{sleep}	Sleeping time	10 - 40 min
$P_{desired}$	Desired percentage of active seeders	10% - $P_{seeders}$

TABLE 1 Simulation parameters.

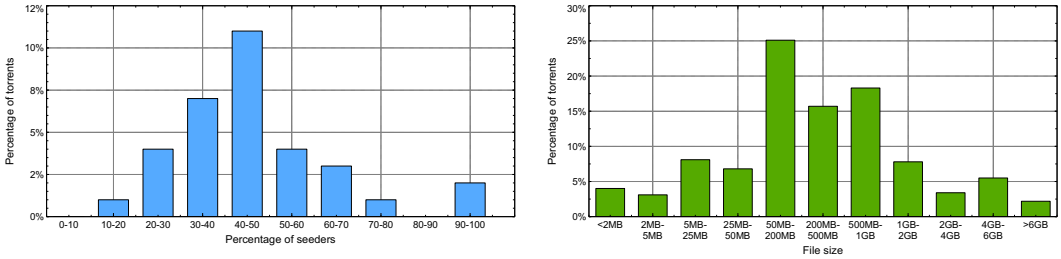
The simulations have been executed with an initial number of peers (N_{peers}) that varies between 1000 and 4000. The number of peers remains almost constant throughout the simulation, because peers join and leave the network at the same rate as suggested in[8]. The percentage of seeders present in the network, $P_{seeders}$, is extracted from[8] and is based on a 23-day analysis of 34 BitTorrent sites with nearly four million users. Figure 4(a) shows how these torrents are grouped according to the percentage of seeders in the network. It can be noted that most of the analyzed sites have a percentage of seeders between 30% and 60%. The size of the shared file, F_{size}, varies from 256 MB to 1024MB in accordance with the most common values of files downloaded in BitTorrent networks[11] (see Figure 4(b)). These two fundamental parameters ($P_{seeders}$ and F_{size}) are in line with the most recent statistics obtained in November 2019 from The Pirate Bay, one of the main torrent search sites. By analysing 697 torrents associated to the 10 most popular TV series obtained from IMDb and indexed by The Pirate Bay, we found that the average percentage of seeders was 55% and the average size of shared file is 850MB.

Each run simulates a time horizon (T_{sim}) of 24 hours. According with[9], the power consumed by a peer in normal mode, p_{high}, and that consumed in sleep mode, p_{low}, are assumed to be 150 and 5 W, respectively. The amounts of time to switch from sleep to normal mode ($T_{sleep\_to\_normal}$) and vice versa ($T_{normal\_to\_sleep}$) are taken from[10]. The check time (see line 13 of Algorithm 1), T_{check}, is equals to 120 seconds. The upload rate of each node, U_{rate}, ranges between 640 and 4096 kbit/s. The sleeping time of seeders (see line 7 of Algorithm 2), T_{sleep}, ranges between 10 and 40 minutes. Specifically, the seeders with the highest upload rate (4096 kbit/s) use the shortest sleeping time (10 minutes), while the seeders with the lowest upload rate (640 kbit/s) use the longest sleeping time (40 minutes). Finally, the desired percentage of active seeders (see line 4 of Algorithm 2), $P_{desired}$, varies from 10% to $P_{seeders}$, for example in a network with 50% of seeders $P_{desired}$, it varies from 10% to 50%. According to the BitTorrent protocol, the number of neighbors known by each node is on average equal to 50. To make the results of the simulations more significant, we repeated the experiments 10 times for each input configuration.

To maintain the percentage of seeders equal to $P_{seeders}$ throughout the simulation, at each run if the current percentage of seeders exceeds $P_{seeders}$ (because some leechers completed the file download and became seeders), some excess seeder are replaced by new leechers. This reproduces the actual behavior of the so-called *transient* seeders that leave the network immediately after completing the download operations[8].

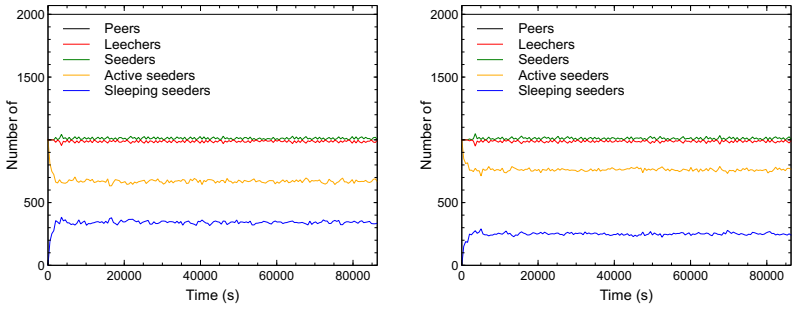
5.3 | Simulation results

Figure 5 shows the evolution of two BitTorrentSW networks composed by 2000 peers with a percentage of seeders equal to 50 and file size of 256 MB. Specifically, Figure 5(a) shows the number of leechers, seeders, active and sleeping seeders over time in BitTorrentSW using $P_{desired} = 30\%$, while Figure 5(b) presents the same numbers using $P_{desired} = 40\%$.



(a) Percentage of seeders available in the network. (b) File size.

FIGURE 4 Torrent distribution in terms of percentage of seeders available in the network (from [8]) and file size (from [11]).



(a) $P_{desired} = 30\%$. (b) $P_{desired} = 40\%$.

FIGURE 5 BitTorrentSW: Number of leechers, seeders, and sleeping seeders over time ($N_{peers} = 2000$, $F_{size} = 256$MB, $P_{seeders} = 50\%$).

Figure 5(a) and Figure 5(b) show that the evolution of the network is very stable. Indeed, in both simulations, the number of peers remains stable around 2000, while the number of peers playing the role of seeders (either sleeping or active) is about 1000 in accordance with the percentage of seeders (50%), and thus the number of leechers is about 1000. There is only a small transient phase (about 5000 seconds) before BitTorrentSW is able to reach a stable percentage of seeders in sleep mode. In Figure 5(a), about 600 (out of 2000 nodes) are active seeders in accordance with the desired percentage of seeders, which is 30%. Similarly, in Figure 5(b), there are about 800 active seeders as the desired percentage of seeders is 40%.

Table 2 shows the total energy consumed by the peers (E_{tot}) and the average download time (D_{avg}) of BitTorrentSW vs. BitTorrent (case indicated as *noSW* in the table). In the case of BitTorrentSW, the network is configured with $N_{peers} = 2000$, $F_{size} = 256$MB, $P_{seeders} = 50\%$ and $P_{desired}$ that varies from 10% to 50%. As one might expect, the lower $P_{desired}$, the higher the energy saving. For instance, using $P_{desired} = 10\%$, 20.6% of energy is saved compared with standard BitTorrent, while with $P_{desired} = 50\%$, the energy saving is 6.2%. On the other hand, energy saving comes with a slight increase in download time. This is because small values of $P_{desired}$ correspond to a lower number of seeders available for download. For instance, using $P_{desired} = 10\%$, there is an increase of 9.8% of the average download time compared to a standard BitTorrent network, while with $P_{desired} = 50\%$, the download time increase is 1.3%.

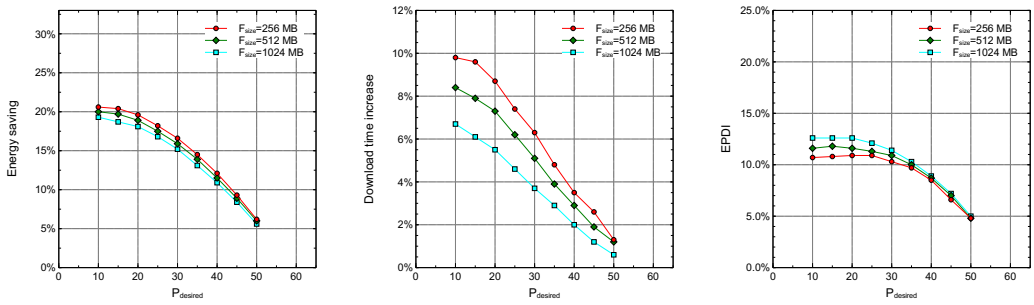
The first two graphs in Figure 6 show the trends of energy saving and download time increase, under the same parameters used in Table 2, but with three different file sizes (256MB, 512MB, 1024MB). The third graph represents the Energy-Download Performance Index (EDPI) defined as follows:

$$EDPI = \frac{E_{tot}^{noSW} - E_{tot}^{SW}}{E_{tot}^{noSW}} - \frac{D_{avg}^{SW} - D_{avg}^{noSW}}{D_{avg}^{noSW}} \quad (3)$$

From its definition, higher EDPI values are preferred to lower ones. By analyzing Figure 6(c), we see that in general EDPI increases by decreasing $P_{desired}$. However, when we use $P_{desired} < 20$, EDPI stops decreasing, which suggests that is not useful to have $P_{desired} < 20$. To fully understand why using lower values of $P_{desired}$ does not result in higher EDPI values, we can take into considerations the graphs in Figure 7, which refer to two file sizes (256 and 1024MB) under the same network conditions ($N_{peers} = 2000$, $P_{seeders} = 50\%$). For each

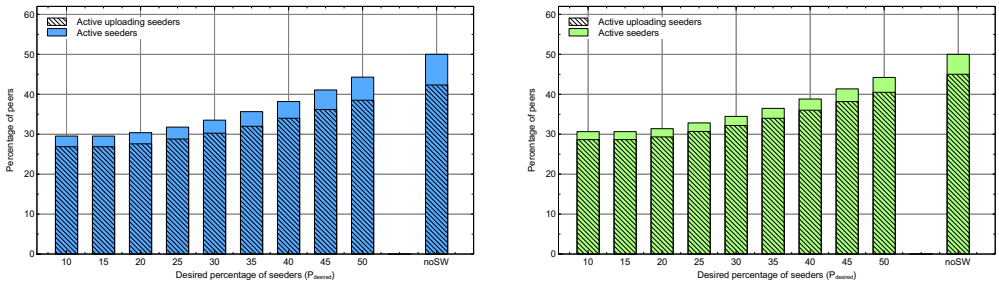
$P_{desired}$	Energy cons. (KW)	Energy saving	Avg download time (sec)	Download time increase
10%	238	+20.6%	2378	+9.8%
15%	239	+20.4%	2373	+9.6%
20%	241	+19.6%	2353	+8.7%
25%	245	+18.2%	2325	+7.4%
30%	250	+16.6%	2301	+6.3%
35%	256	+14.5%	2268	+4.8%
40%	264	+12.1%	2241	+3.5%
45%	272	+9.3%	2221	+2.6%
50%	281	+6.2%	2193	+1.3%
noSW	*300*	*0%*	*2165*	*0%*

TABLE 2 Total energy consumption and average download time of BitTorrentSW vs. BitTorrent with $P_{desired}$ ranging from 10% to 50% ($N_{peers} = 2000$, $F_{size} = 256$MB, $P_{seeders} = 50\%$)



(a) Energy saving. (b) Download time increase. (c) EDPI.

FIGURE 6 Energy saving, download time increase and EDPI of BitTorrentSW considering three file sizes (F_{size}), with $P_{desired}$ ranging from 10% to 50% ($N_{peers} = 2000$, $P_{seeders} = 50\%$)



(a) $F_{size} = 256$MB. (b) $F_{size} = 1024$MB.

FIGURE 7 Percentage of active seeders and uploading active seeders, with $P_{desired}$ ranging from 10% to 50% ($N_{peers} = 2000$, $P_{seeders} = 50\%$).

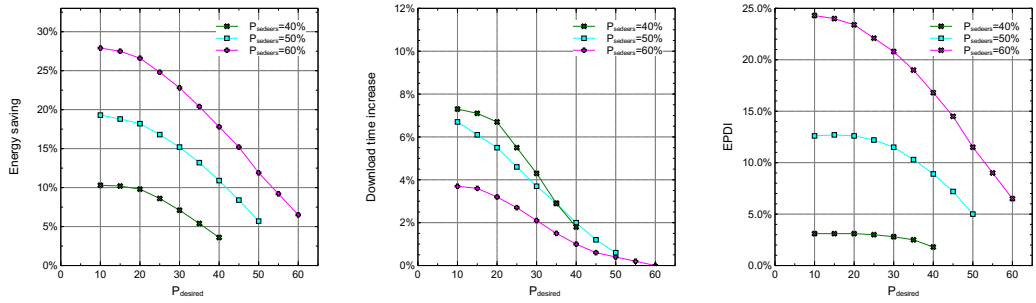
value of $P_{desired}$, the two graphs show the percentage of active seeders, as well as the percentage of seeders that are currently performing upload operations. It can be observed that the difference between the two percentages (i.e., the percentage of active seeders that are not performing any upload operation) remain constant when $P_{desired} \leq 20$. Since BitTorrentSW is able to switch to sleep mode only seeders not performing uploads, we conclude that using $P_{desired} < 20$ does not improve EDPI.

Figure 8 shows the trends of energy saving, download time increase and EDPI with $P_{desired}$ that varies from 10% to $P_{seeders}$, and three different values of $P_{seeders}$ (40%, 50%, 60%). Figures 8(a) and 8(b) shows that the higher the percentage of seeders present in the network the greater the energy that BitTorrentSW manages to save and the lower the increase in average download times. Figure 8(c) shows that

EDPI increases by decreasing $P_{desired}$, particularly for the higher values of $P_{desired}$. Figures 9(a) and Figure 9(b), which refer to $P_{seeders} = 40\%$ $P_{seeders} = 60\%$, show that the percentage of active seeders that are not performing any upload operation remain constant when $P_{desired} \leq 30$ and $P_{desired} \leq 15$, respectively.

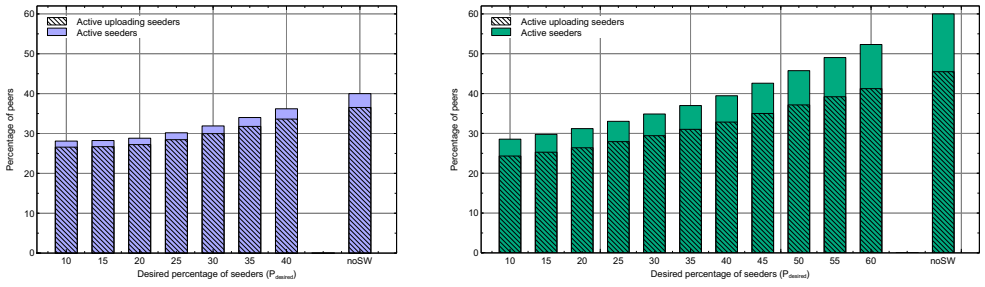
Figure 10(a) shows the total energy saved (in KW) by BitTorrentSW, when the number of peers increases from 1000 to 4000, for three values of $P_{desired}$. As shown by the histograms, the energy saved by BitTorrentSW grows linearly with the number of nodes in all the considered configurations (different $P_{desired}$ values). Figure 10(b) shows how the average download time varies by increasing the network size. As shown by the histograms, when the network size increases, the download time slightly decreases. This is because the chance of finding the missing chunks increases as the network grows.

In summary, the simulation results show that the BitTorrentSW approach is effective in reducing energy consumption without significantly affecting download time. Moreover, BitTorrentSW scales well, as it ensures that energy saving is proportional to network size, and download time does not increase as the network becomes larger.



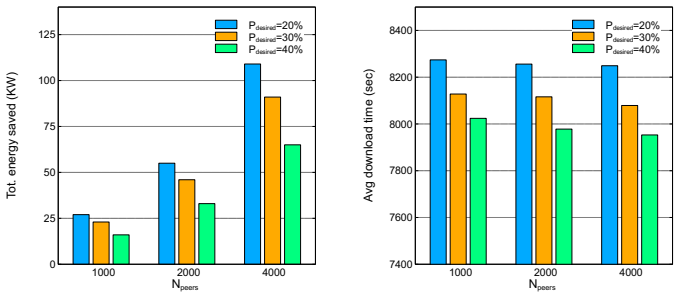
(a) Energy saving. (b) Download time increase. (c) EDPI.

FIGURE 8 Energy saving, download time increase and EDPI of BitTorrentSW considering three percentages of seeders ($P_{seeders}$), with $P_{desired}$ ranging from 10% to 50% ($N_{peers} = 2000$, $F_{size} = 1024$MB)



(a) $P_{seeders} = 40\%$. (b) $P_{seeders} = 60\%$.

FIGURE 9 Percentage of active seeders and uploading active seeders, with $P_{desired}$ ranging from 10% to $P_{seeders}$ ($N_{peers} = 2000$, $F_{size} = 1024$MB).



(a) Tot. energy saved. (b) Avg download time.

FIGURE 10 Total energy saved and average download time of BitTorrentSW when N_{peers} increases from 1000 to 4000, for three values of $P_{desired}$ ($P_{seeders} = 50\%$, $F_{size} = 1024MB$).

6 | CONCLUSIONS

Reducing energy consumption in distributed systems is a challenging task, as it involves design and optimization of energy-aware algorithms, architectural models, and applications. This is particularly true in large-scale peer-to-peer networks, given the need of obtaining significant energy savings without affecting the performance perceived by the final users [29].

In this paper we focused on BitTorrent, one of the most popular communication protocol for peer-to-peer networks, by proposing *BitTorrentSW*, a sleep-and-wake approach for saving energy by ensuring good file sharing performance. In BitTorrentSW each seeder autonomously decides if and when switching to sleep mode according to the local information it owns about network composition: if the percentage of seeders in the network exceeds a certain threshold, the seeder switches to sleep mode for a period of time. Differently from other solutions in the literature, BitTorrentSW does not modify the BitTorrent protocol with the introduction of new messages. In fact, when BitTorrentSW let a peer switch to sleep mode, the sleeping peer is seen by all the other peers as if it has left the network, and thus no additional actions must be carried out. Given the generality of the approach and the possibility to use it without modifying the standard protocols of a P2P network, the sleep-and-wake technique proposed in this manuscript may be exploited in other P2P systems.

BitTorrentSW has been evaluated through PeerSim using parameters extracted from real BitTorrent traces. The simulation results show that, in all the configurations under analysis, the percentage of energy saved by BitTorrentSW is much higher than the percentage of increase in download time. For instance, in a network with 50% of seeders, about 20% of energy is saved using BitTorrentSW, with an increase of 7% of the average time needed to complete a file download compared to a standard BitTorrent network in which all seeders are always powered on. The results are further improved when the percentage of seeders in the network increases. For example, with 60% of seeders, we registered 28% of energy saving with only 4% of increase in download time. To summarize, all the results demonstrate the effectiveness of BitTorrentSW in reducing energy consumption with a negligible impact on the average download time.

ACKNOWLEDGMENT

This work has been partially supported by the ASPIDE Project funded by the European Union Horizon 2020 research and innovation programme under grant agreement No 801091.

References

1. Malatras A, Peng F, Hirsbrunner B. Energy-efficient peer-to-peer networking and overlays. *Handbook of Green Information and Communication Systems, Elsevier.* 2013;:513–540.

2. The Global Internet Phenomena Report, Sandvine, 2019 .

3. Blackburn Jeremy, Christensen Ken. A simulation study of a new green bittorrent. In: :1–6IEEE; 2009.

4. Forshaw Matthew, Thomas Nigel. A novel approach to energy efficient content distribution with BitTorrent. In: :188–196Springer; 2012.

5. Lee Yong-Ju, Jeong Jin-Hwan, Kim Hag-Young, Lee Cheol-Hoon. Energy-saving set-top box enhancement in BitTorrent networks. In: :809–812IEEE; 2010.

6. Montresor Alberto, Jelasity Márk. PeerSim: A Scalable P2P Simulator. In: :99-100; 2009; Seattle, WA.

7. Marozzo Fabrizio, Marzano Francesco, Talia Domenico, Trunfio Paolo. BitTorrentSW: A Sleep-and-Wake approach to reduce energy consumption in BitTorrent networks. In: :144-150; 2018; Orléans, France.

8. Bieber Justin, Kenney Michael, Torre Nick, Cox Landon P. An empirical study of seeders in BitTorrent. *Technical Report CS 2006–08*. 2006;.

9. Setz Brian, Nizamic Faris, Lazovik Alexander, Aiello Marco. Power management of personal computers based on user behaviour. In: :1–8IEEE; 2016.

10. Agarwal Yuvraj, Hodges Steve, Chandra Ranveer, Scott James, Bahl Paramvir, Gupta Rajesh. Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage.. In: :365–380; 2009.

11. Cheng Jie, Donahue Ryder. The Pirate Bay Torrent Analysis and Visualization. *International Journal of Science, Engineering and Computer Technology*. 2013;3(2):38.

12. Orgerie Anne-Cecile, Assuncao Marcos Dias de, Lefevre Laurent. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys (CSUR)*. 2014;46(4):47.

13. Khan Samee U, Ahmad Ishfaq. A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. *IEEE Transactions on Parallel and Distributed Systems*. 2008;20(3):346–360.

14. Mi Haibo, Wang Huaimin, Yin Gang, Zhou Yangfan, Shi Dianxi, Yuan Lin. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: :514–521IEEE; 2010.

15. Purushothaman Pradeep, Navada Mukund, Subramaniyan Rajagopal, Reardon Casey, George Alan D. Power-proxying on the NIC: a case study with the Gnutella file-sharing protocol. In: :519–520IEEE; 2006.

16. Ripeanu Matei. Peer-to-peer architecture case study: Gnutella network. In: :99–100IEEE; 2001.

17. Anastasi Giuseppe, Giannetti Ilaria, Passarella Andrea. A BitTorrent proxy for Green Internet file sharing: Design and experimental evaluation. *Computer Communications*. 2010;33(7):794–802.

18. Enokido Tomoya, Aikebaier Ailixier, Takizawa Makoto. A model for reducing power consumption in peer-to-peer systems. *IEEE Systems Journal*. 2010;4(2):221–229.

19. Kelényi Imre, Nurminen Jukka K. Optimizing energy consumption of mobile nodes in heterogeneous Kademlia-based distributed hash tables. In: :70–75IEEE; 2008.

20. Maymounkov Petar, Mazieres David. Kademlia: A peer-to-peer information system based on the xor metric. In: :53–65Springer; 2002.

21. Leung Andrew Ka-Ho, Kwok Yu-Kwong. On localized application-driven topology control for energy-efficient wireless peer-to-peer file sharing. *IEEE Transactions on Mobile Computing*. 2008;7(1):66–80.

22. Han Jung-Suk, Song Jin-Woo, Kim Taek-Hun, Yang Song-Bong. Double-layered mobile P2P systems using energy-efficient routing schemes. In: :122–127IEEE; 2008.

23. Joseph Mary Suchitha, Kumar Mohan, Shen Huaping, Das Sajal. Energy efficient data retrieval and caching in mobile peer-to-peer networks. In: :50–54IEEE; 2005.

24. Park Kwangjin, Valduriez Patrick. Energy efficient data access in mobile p2p networks. *IEEE Transactions on Knowledge and Data Engineering*. 2011;23(11):1619–1634.

25. Tung Yu-Chih, Lin Kate Ching-Ju. Location-assisted energy-efficient content search for mobile peer-to-peer networks. In: :477–482IEEE; 2011.

26. Andrew Lachlan LH, Sucevic Andrew, Nguyen Thuy TT. Balancing peer and server energy consumption in large peer-to-peer file distribution systems. In: :76–81IEEE; 2011.

27. Sucevic Andrew, Andrew Lachlan LH, Nguyen Thuy TT. Powering down for energy efficient peer-to-peer file distribution. *ACM SIGMETRICS Performance Evaluation Review*. 2011;39(3):72–76.

28. Corigliano Salvatore, Trunfio Paolo. Exploiting Sleep-and-Wake Strategies in the Gnutella Network. In: :406–412IEEE Computer Society Press; 2014; Minneapolis, USA. ISBN 978-1-4799-5158-1.

29. Trunfio Paolo. A two-layer model for improving the energy efficiency of file sharing peer-to-peer networks. *Concurrency and Computation: Practice and Experience*. 2015;27(13):3166—3183.

How to cite this article: F. Marozzo, D. Talia, and P. Trunfio (2019), A Sleep-and-Wake technique for reducing energy consumption in BitTorrent networks, *Concurrency and Computation: Practice and Experience*, *2020;xx:x–x*.