BitTorrentSW: A Sleep-and-Wake Approach to Reduce Energy Consumption in BitTorrent Networks

Conference Paper · July 2018

DOI: 10.1109/HPCS.2018.00037

4 authors, including:

Fabrizio Marozzo
Università della Calabria
60 PUBLICATIONS **483** CITATIONS

SEE PROFILE

Domenico Talia
Università della Calabria
407 PUBLICATIONS **5,113** CITATIONS

SEE PROFILE

BitTorrentSW: A Sleep-and-Wake approach to reduce energy consumption in BitTorrent networks

Fabrizio Marozzo, Francesco Marzano, Domenico Talia, Paolo Trunfio
DIMES Department
University of Calabria
Rende, Italy
Email: [fmarozzo, fmarzano, talia, trunfio]@dimes.unical.it

***Abstract*—File sharing over peer-to-peer networks has been one of the most important Internet applications over the past twenty years. Given the very large number of hosts involved in peer-to-peer networks, reducing their aggregate energy consumption is an important challenge to be faced. In this paper, we study how the sleep-and-wake energy saving approach can be used to reduce energy consumption in BitTorrent, one of the most popular file sharing peer-to-peer networks. We introduce BitTorrentSW, a sleep-and-wake approach for BitTorrent networks that allows peers to cyclically switch between wake and sleep mode to save energy while ensuring good file sharing performance. The decision to get in sleep mode is taken independently by each peer 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 about 40% of energy is saved using BitTorrentSW, with only an increase of 5% of the average time needed to complete a file download compared to a standard BitTorrent network in which all peers are always powered on.**

***Index Terms*—Peer-to-peer, BitTorrent, File sharing, Energy efficiency, Performance analysis**

I. Introduction

As peer-to-peer networks gather and share large sets of hosts, their aggregate energy consumption has become an important challenge to be addressed. The importance of the problem has led several researchers to propose solutions for improving the energy efficiency of peer-to-peer networks. Common approaches toward this goal include the use of proxies, optimizing task allocation, message reduction, location-based mechanisms, overlay structure optimization, and the "sleep-and-wake" strategy [15]. The latter is one of the most important approaches, relying on the fact 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 active.

In this paper, we study how the sleep-and-wake energy saving approach can be used to reduce energy consumption in BitTorrent, the most popular peer-to-peer networks for transferring digital contents among users, accounting for more than half of total file sharing bandwidth and 2.26% of global Internet traffic during 2013 [12].

We introduce *BitTorrentSW*, a sleep-and-wake approach for BitTorrent networks that allows peers to cyclically switch between wake and sleep mode to save energy while ensuring good file sharing performance. In BitTorrent, peers can be *seeders* or *leechers*: the formers hold complete files and share them; the latters are downloading the parts they need to complete the file, and share the parts they already have. In BitTorrentSW only the seeders can get in 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 can get in 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 [4][8][13], but they differ from BitTorrentSW in three main aspects: 1) 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; 2) 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; 3) some of them assume that PCs 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.

BitTorrentSW has been evaluated through PeerSim using real BitTorrent traces. The simulation results show that about 40% of energy is saved using BitTorrentSW, with only an increase of 5% of the average time needed to complete a file download compared to a standard BitTorrent network in which all peers are always powered on. This result is very positive as it is significantly reduced the energy consumption with a negligible increase in average download time.

The remainder of the paper is structured as follows. Section II discusses related work. Section III presents the system model. Section IV describes the BitTorrentSW algorithm. Section V presents an evaluation of the energy-saving algorithm using PeerSim. Finally, Section VI concludes the paper.

II. Related Work

Malatras et al. [15] classified existing solutions for improving the energy-efficient of peer-to-peer networks in six categories:

1) The *proxying approach* is based on the use by peers of proxies to delegate some of their activities, such as file

downloading. Using proxies, peer-to-peer hosts do not need to stay constantly online, this way reducing the overall energy consumption. Examples of proxy-based approaches are the system proposed by Anastasi et al. [2] for reducing the energy consumption of hosts running the BitTorrent application, and the system by Purushothaman et al. [19] for Gnutella networks [20].

2) *Task allocation optimization* is based on the observation that significant energy savings can be achieved in a peer-to-peer network 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. [6][7], 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 minimizing the number of messages exchanged through the peer-to-peer network with the goal of 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 [11], who adopted a selective message dropping mechanism for reducing the number of messages exchanged in a Kademlia network [17].
4) The *location-based approach* exploits positioning information about nodes to make 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 [10], Park and Valduriez [18], and Tung and Lin [24].
5) *Overlay structure optimization* aims at improving the energy efficiency of a peer-to-peer network by either controlling its topology during construction or maintenance, or introducing new layers to the overlay. An example of the first type is the work by Leung and Kwok [14], 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. [9].
6) The *sleep-and-wake approach* aims at reducing the overall energy consumption of a peer-to-peer network by letting 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, in order to avoid excessive degradation of system performance. Several systems fall in this category, including the ones by Andrew et al. [3], Sucevic et al. [22], Corigliano et al. [5] and Trunfio [23].

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 [4] proposed a BitTorrent extension called *green BitTorrent*. This solution tries to optimize energy consumption through a sleep-and-wake approach allowing peers to go into sleep mode when they are not downloading/uploading chunks, but keeping the peers 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 that is not doing upload/download activities; ii) it also modifies the BitTorrent protocol by introducing new messages related to the awakening of the peers; iii) it finally assumes that PCs are equipped with a Wake-on-LAN connector which allows the peer to be re-awakened with a message. Our approach puts in 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.

Lee et al. [13] proposed a solution based on the definition of new states for the peers involved in the BitTorrent network. 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 (Wake on LAN) technology, re-establishing the TCP connections. A main difference with our approach is that the solution of 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 the seeders can decide autonomously, based on local information, whether to go into energy saving mode or not.

Forshaw and Thomas [8] proposed an energy saving solution for BitTorrent networks. Basically they consider a seed pool, that is a group of servers waiting to share content to peers of the network, which guarantees satisfactory levels 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 in order to reduce 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.

III. System model and design principles

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

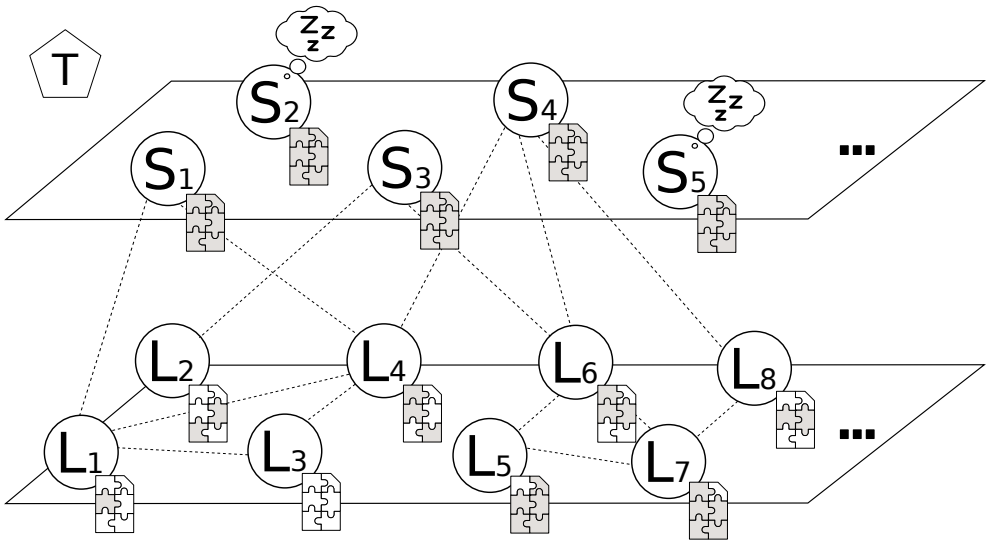


Fig. 1. An example of BitTorrent network for a file F composed by a tracker T, a set of seeders $(S_1 \dots S_5)$ and a set of lechers $(L_1 \dots L_8)$.

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 for a file F. In this example, F is divided into six chunks. A set of *seeders*, $S_1 \dots S_5$ own the complete file, and a set of *leechers*, $L_1 \dots L_8$ are trying to complete the download of F. A tracker T keeps track of the leechers that are downloading the file and 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.

The *BitTorrentSW* approach is based on the idea of keeping in sleep mode a subset of the 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 getting in sleep mode 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 can get in sleep mode for a period of time. Details on the sleep-and-wake algorithm will be provided in the next section.

IV. 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 viceversa. 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 3 shows the state diagram that describes the behavior of a peer of 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 the 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.Sleep*, which is a seeder in sleep mode. If a seeder is not performing any upload operation, it can change from *Seeder.Active* to *Seeder.Sleep*. The decision is made autonomously by each seeder according to the local information that it has on the network: if the percentage of seeders present in the network (*currentSeedPerc*) exceeds a certain threshold (*desideredPerc*), the seeder can go sleeping for

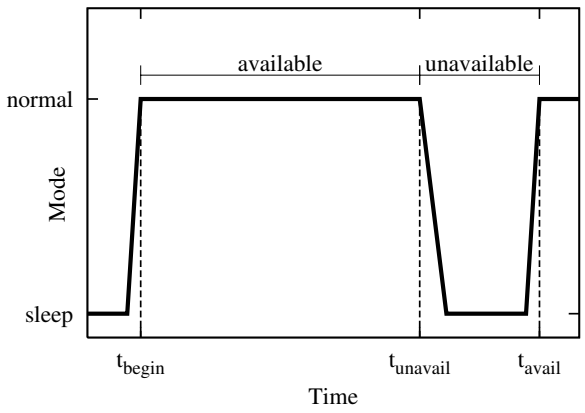


Fig. 2. Relationship between availability status and power mode of a seeder.

a certain amount of time (*sleepingTime*). After *sleepingTime*, the seeder will return to be active (*Seeder.Active*).

Figure 4 shows the BitTorrentSW power management algorithm, which implements the strategy outlined above. A peer cyclically executes the power management method (lines 1-15). If the peer is an active seeder and it is not performing any upload operation (line 2), it calculates the current percentage of seeders according to local information (line 3). This is done by a localPercSeeders() function, which returns the percentage of this seeder's neighbors that possess all the file chunks (this info is available to every BitTorrent node). Then, the seeder calculates the difference between *currentPercSeeder* and the desidered 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). In this case, the seeder calculates a random number between 0 and *currentPercSeeder* (line 6). If this random value is lower than *difference* (line 7), the status of the seeder is changed to *Seeder.Sleep* (line 8) and the seeder gets in sleep mode for a given *sleepingTime* (line 9). After the sleeping time, the seeder wakes up and returns to be active (line 10). The cycle is repeated after *checkTime* seconds (line 14).

```
// executed by every peer
P_i.power_management()
 1: while true do
 2:     if status = Seeder.Active and isNotUploading then
 3:         currentPercSeeder := localPercSeeders();
 4:         difference := currentPercSeeder - desiredPerc;
 5:         if difference > 0 then
 6:             random := randomNumber(0, currentPercSeeder);
 7:             if random < difference then
 8:                 status := Seeder.Sleep;
 9:                 go in sleep mode for sleepingTime seconds;
10:                 status := Seeder.Active;
11:             end if
12:         end if
13:     end if
14:     wait for checkTime seconds;
15: end while
```

Fig. 4. BitTorrentSW power management algorithm.

V. Performance Evaluation

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

A. 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 for sharing a file F composed of N_{peers} peers is created; a percentage $P_{leechers}$ of these peers is selected to play the role of leechers, while the remaining percentage $P_{seeders}$ of peers is selected to act as seeders.
2) To each seeder is assigned 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 $C_{size}) = 256kB$ is the standard chunk size in BitTorrent.
3) The simulation begins, with seeders performing server-side activities (file uploads) and leechers performing both client-side activities (file downloads) and server-side activities (file uploads).

The simulator assumes that seeders are not used by other applications, and therefore they can put in sleep mode taking into account only their upload activities on file 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:

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

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;

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

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;

B. Simulation parameters

Table I reports the parameters used for the simulations. The most relevant parameters are extracted from real BitTorrent traces presented in [26] and [25].

The network has a number of peers N_{peers} equal to 1000, the 30% of which initially configured to play the role of seeders. To simulate network dynamism, a joining rate J_{rate} and a leaving rate L_{rate} have been defined. On average, every minute J_{rate} leechers join the network (new peers interested in downloading the file), while L_{rate} seeders leave the network (peers that leave the network after obtaining the file). In our simulations J_{rate} = L_{rate} to keep the total number of nodes approximatively constant during each run, following the approach described in [16]. Each run simulates a time horizon (T_{sim}) of 24 hours. According with [21], the power consumed by a host 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

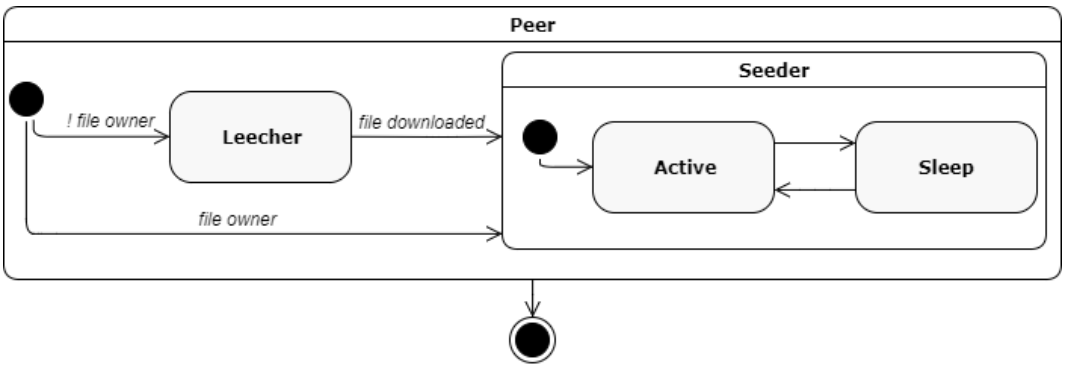


Fig. 3. State diagram of a BitTorrentSW peer.

TABLE I
SIMULATION PARAMETER

Parameter	***Description***	***Values***
N_{peers}	Number of nodes in the network	1000
$P_{seeders}$	Initial percentage of seeders	0.3
$P_{leechers}$	Initial percentage of leechers	0.7
J_{rate}	Joining rate: avg. number of nodes that join the network every minute (% on N_{peers})	0.5
L_{rate}	Leaving rate: avg. number of nodes that leave the network every minute (% on N_{peers})	0.5
T_{sim}	Simulation time	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	60 s
F_{size}	Size of the shared file	1024 MB
U_{rate}	Upload rate	640 - 4096 Kb/s
T_{sleep}	Sleeping time	10 - 40 min
$P_{desidered}$	Desired percentage of active seeders	10% - 50%

vice versa ($T_{normal\_to\_sleep}$) are taken from [1]. The check time (see line 14 of Figure 4), T_{check}, is equals to 60 seconds. The size of the shared file, F_{size}, is equal to 1024 MB. The upload rate of each node, U_{rate}, ranges between 640 and 4096 kbit/s. The sleeping time of seeders (see line 9 of Figure 4), 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 Figure 4), $P_{desired}$, varies from 10% to 50%.

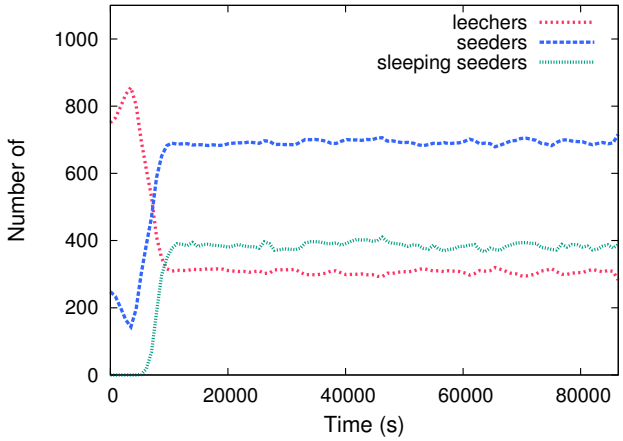
C. Simulation results

Figure 5 compares the evolution of BitTorrentSW and BitTorrent networks under the same simulation parameters, except for the the sleeping time which is null for the standard BitTorrent network. Specifically, Figure 5(a) shows the number of leechers, seeders, and sleeping seeders over time in BitTorrentSW using $P_{desired} = 30\%$, while Figure 5(b) presents the same numbers obtained in a standard BitTorrent network (in this case, the number of sleeping seeders is 0).

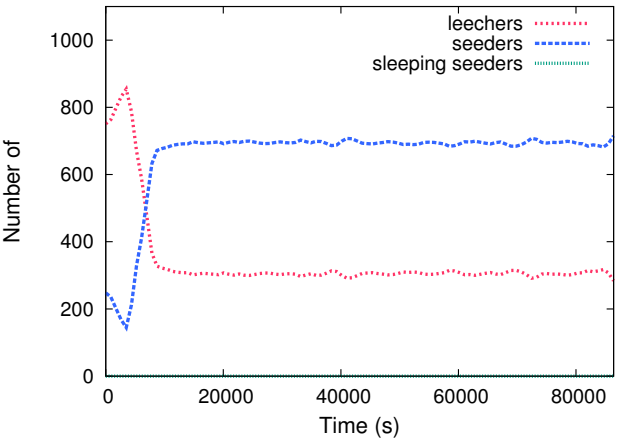
Both Figure 5(a) and Figure 5(b) show that there are two phases in the evolution of the network. In the first phase of the simulation (approximatively, the first 10000 seconds), there is a reduction in the number of seeders and an increase in the number of leechers. This is because the frequency of incoming nodes (leechers) is greater than the frequency at which leechers are transformed into seeders. After this transitory phase, the network reaches its steady state, with about 30% of peers playing the role of leechers and 70% acting as seeders (either sleeping or active). With BitTorrentSW (see Figure 5(a)), about 40% of peers are sleeping seeders, and therefore about 30% of peers are active seeders, which corresponds to the desired percentage of active seeders in the network ($P_{desired} = 30\%$). With standard BitTorrent (see Figure 5(b)), the number of sleeping seeders is zero all over the simulation.

Figure 6 shows the total energy consumed by the network (E_{tot}) and the average download time (D_{avg}) of BitTorrentWS vs. BitTorrent. In case of BitTorrentSW, three different values of $P_{desired}$ are considered: 10%, 30% and 50%.

Figure 6(a) shows the total energy consumed by the network. With standard BitTorrent (case indicated as *noWS* in the

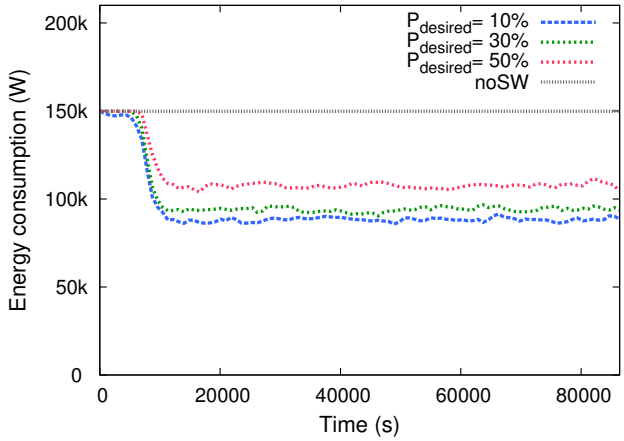


(a) BitTorrentSW.

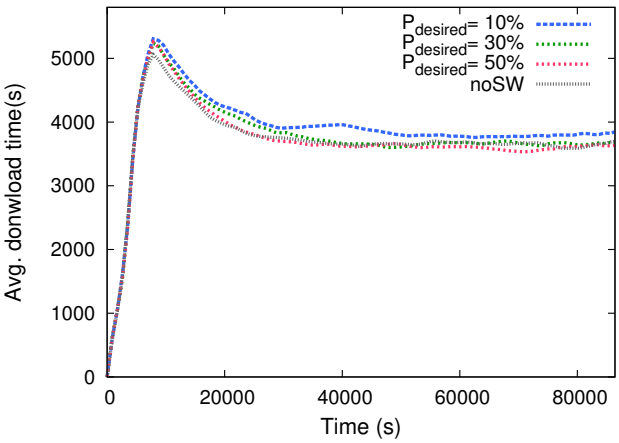


(b) BitTorrent.

Fig. 5. BitTorrentSW vs. BitTorrent: Number of leechers, seeders, and sleeping seeders over time.



(a) Total energy consumed by the network.



(b) Average download time.

Fig. 6. BitTorrentSW vs. BitTorrent (noSW case): Total energy consumed and average download time.

graph), the energy consumed is constant during the entire simulation, because the seeders are always powered on. With BitTorrentSW, after a first phase of about 10000 seconds (as already commented for Figure 5), the energy consumption decreases by using lower values of $P_{desired}$. In fact, the lower $P_{desired}$, the higher the number of sleeping seeders, and therefore the lower the total energy consumed by the network. For instance, using $P_{desired} = 10\%$, about 40% of energy is saved compared with standard BitTorrent.

Figure 6(b) shows the average download time. During the first phase, the download time increases because there is a high number of leechers and a low number of seeders. When the network reaches its steady state, the average download time decreases and remains constant until the end of the simulation. The download time increases slightly passing from standard BitTorrent (*noWS* case) to decreasing $P_{desired}$ values. For example, with the most energy saving configuration ($P_{desired} = 10\%$), there is only an increase of 5% of the average download time compared to a standard BitTorrent network in which all peers are always powered on.

In summary, the simulation results show that the BitTorrentSW approach is effective in reducing energy consumption without significantly affecting download time.

VI. 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 necessity of obtaining significant energy savings without affecting the performance perceived by the final users[23].

In this paper we focused on one of the most popular peer-to-peer networks, by proposing *BitTorrentSW*, a sleep-and-wake approach for BitTorrent networks that allows peers to cyclically switch between wake and sleep mode to save energy while ensuring good file sharing performance. In BitTorrentSW each seeder autonomously decides if and when get in 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 can get in 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.

BitTorrentSW has been evaluated through PeerSim using real BitTorrent traces. The simulation results show that about 40% of energy is saved using BitTorrentSW, with only an increase of 5% of the average time needed to complete a file download compared to a standard BitTorrent network in which all peers are always powered on. This result is very positive as it is significantly reduced the energy consumption with a negligible increase in average download time.

REFERENCES

[1] Yuvraj Agarwal, Steve Hodges, Ranveer Chandra, James Scott, Paramvir Bahl, and Rajesh Gupta. Somniloquy: Augmenting network interfaces to reduce pc energy usage. In *NSDI*, volume 9, pages 365–380, 2009.

[2] Giuseppe Anastasi, Ilaria Giannetti, and Andrea Passarella. A bittorrent proxy for green internet file sharing: Design and experimental evaluation. *Computer Communications*, 33(7):794–802, 2010.

[3] Lachlan LH Andrew, Andrew Sucevic, and Thuy TT Nguyen. Balancing peer and server energy consumption in large peer-to-peer file distribution systems. In *Online Conference on Green Communications (GreenCom), 2011 IEEE*, pages 76–81. IEEE, 2011.

[4] Jeremy Blackburn and Ken Christensen. A simulation study of a new green bittorrent. In *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[5] Salvatore Corigliano and Paolo Trunfio. Exploiting sleep-and-wake strategies in the gnutella network. In *Proc. of the 15th Int. Conference on Collaboration Technologies and Systems (CTS 2014)*, pages 406–412, Minneapolis, USA, 19-23 May 2014. IEEE Computer Society Press. ISBN 978-1-4799-5158-1.

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

[7] Tomoya Enokido, Kota Suzuki, Ailixier Aikebaier, and Makoto Takizawa. Laxity based algorithm for reducing power consumption in distributed systems. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on*, pages 321–328. IEEE, 2010.

[8] Matthew Forshaw and Nigel Thomas. A novel approach to energy efficient content distribution with bittorrent. In *European Workshop on Performance Engineering*, pages 188–196. Springer, 2012.

[9] Jung-Suk Han, Jin-Woo Song, Taek-Hun Kim, and Song-Bong Yang. Double-layered mobile p2p systems using energy-efficient routing schemes. In *Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian*, pages 122–127. IEEE, 2008.

[10] Mary Suchitha Joseph, Mohan Kumar, Huaping Shen, and Sajal Das. Energy efficient data retrieval and caching in mobile peer-to-peer networks. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 50–54. IEEE, 2005.

[11] Imre Kelényi and Jukka K Nurminen. Optimizing energy consumption of mobile nodes in heterogeneous kademlia-based distributed hash tables. In *Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST'08. The Second International Conference on*, pages 70–75. IEEE, 2008.

[12] Jonathan F. Lee. Purchase, pirate, publicize: Private-network music sharing and market album sales. *Information Economics and Policy*, 2018.

[13] Yong-Ju Lee, Jin-Hwan Jeong, Hag-Young Kim, and Cheol-Hoon Lee. Energy-saving set-top box enhancement in bittorrent networks. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 809–812. IEEE, 2010.

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

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

[16] Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. P2p-mapreduce: Parallel data processing in dynamic cloud environments. *Journal of Computer and System Sciences*, 78(5):1382–1402, September 2012.

[17] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *International Workshop on Peer-to-Peer Systems*, pages 53–65. Springer, 2002.

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

[19] Pradeep Purushothaman, Mukund Navada, Rajagopal Subramaniyan, Casey Reardon, and Alan D George. Power-proxying on the nic: a case study with the gnutella file-sharing protocol. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 519–520. IEEE, 2006.

[20] Matei Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 99–100. IEEE, 2001.

[21] Brian Setz, Faris Nizamic, Alexander Lazovik, and Marco Aiello. Power management of personal computers based on user behaviour. In *Smart Cities and Green ICT Systems (SMARTGREENS), 2016 5th International Conference on*, pages 1–8. IEEE, 2016.

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

[23] Paolo Trunfio. A two-layer model for improving the energy efficiency of file sharing peer-to-peer networks. *Concurrency and Computation: Practice and Experience*, 27(13):3166–3183, 10 September 2015.

[24] Yu-Chih Tung and Kate Ching-Ju Lin. Location-assisted energy-efficient content search for mobile peer-to-peer networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 477–482. IEEE, 2011.

[25] Boxun Zhang, Alexandru Iosup, and Dick Epema. The peer-to-peer trace archive: Design and comparative trace analysis. technical report pds-2010-003. Technical Report PDS-2010-003, Delft University of Technology Parallel and Distributed Systems Report Series.

[26] Boxun Zhang, Alexandru Iosup, Johan Pouwelse, and Dick Epema. The peer-to-peer trace archive: design and comparative trace analysis. In *Proceedings of the ACM CoNEXT Student Workshop*, page 21. ACM, 2010.