Exploiting Sleep-and-Wake Strategies in the Gnutella Network

Salvatore Corigliano
DIMES, University of Calabria
Rende (CS), Italy
Email: corigliano@si.dimes.unical.it

Paolo Trunfio
DIMES, University of Calabria
Rende (CS), Italy
Email: trunfio@dimes.unical.it

***Abstract*—File sharing over peer-to-peer networks has been one of the most important Internet applications over the past fifteen years, with a vast number of users and nodes participating to these networks every day. Given the large sets of computing resources involved in peer-to-peer file sharing networks, their aggregate energy consumption is an important problem to be addressed, considered the economic and environmental impact of energy production and use. In this paper we evaluate how the sleep-and-wake energy-saving approach can be used to reduce energy consumption in Gnutella, one of the most popular peer-to-peer file sharing networks. In order to save energy, we introduce a general sleep-and-wake algorithm that allows leaf-peers of the Gnutella network cyclically switch between wake and sleep mode, where the time passed in sleep mode is autonomously decided by each leaf-peer. We define different strategies that a leaf-peer may employ to decide the duration of its sleep periods. Such strategies are evaluated through simulation using the general sleep-and-wake algorithm in different network scenarios.**

***Keywords*—Peer-to-peer; File sharing; Gnutella; Energy efficiency; Simulation analysis.**

I. INTRODUCTION

File sharing over peer-to-peer networks has been one of the most important Internet applications over the past fifteen years, with a vast number of users and nodes participating to these networks every day. The latter is also witnessed by the impact of peer-to-peer file sharing on the overall Internet traffic, which some studies have quantified between 40 and 70 percent [1][2]. Given the large sets of computing resources involved in peer-to-peer file sharing networks, their aggregate energy consumption is an important problem to be addressed, considered the economic and environmental impact of energy production and use [3].

The importance of the problem has stimulated several researches aimed at 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 [4]. The latter is one of the most popular approaches, based on the principle that the overall energy consumption of a peer-to-peer system can be significantly reduced if peers periodically switch from *wake* mode (high-power) to *sleep* mode (low-power), and viceversa.

In this paper we evaluate how the sleep-and-wake energy-saving approach can be used to reduce energy consumption in Gnutella [5], one of the most popular peer-to-peer file sharing networks. In order to save energy, we introduce a general sleep-and-wake algorithm that allows leaf-peers of the Gnutella network cyclically switch between wake and sleep mode, where the time passed in sleep mode is autonomously decided by each leaf-peer. We define different strategies that a leaf-peer may employ to decide the duration of its sleep periods. Such strategies are evaluated through simulation using the general sleep-and-wake algorithm in different network scenarios.

The remainder of the paper is structured as follows. Section II discusses related work. Section III describes the network assumptions. Section IV introduces the general sleep-and-wake algorithm. Section V proposes a set of strategies for deciding the sleep duration of a leaf-peer. Section VI evaluates the proposed strategies through simulation. Finally, Section VII concludes the paper.

II. RELATED WORK

Existing research works on energy-efficient peer-to-peer systems can be classified under six categories, according to the approach they follow in reducing energy consumption [4]: proxying, task allocation optimization, message reduction, location-based, overlay structure optimization, and sleep-and-wake.

The proxying approach is based on the use, by peer-to-peer hosts, 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. [6] for reducing the energy consumption of hosts running the BitTorrent application [7], and the system by Purushothaman et al. [8] for the Gnutella network.

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. [9][10], 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.

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 [12].

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 [13], Park and Valduriez [14], and Tung and Lin [15].

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 [16], 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. [17].

Finally, the sleep-and-wake approach aims at reducing the overall energy consumption of a peer-to-peer network by letting peers cyclically switch between wake and sleep mode. Notable systems falling in this category are the ones by Lefebvre and Feeley [18], Blackburn and Christensen [19], Lee et al. [20], Gurun et al. [21], Sucevic et al. [22], Jourjon et al. [23], Andrew et al. [24], and Hlavacs et al. [25].

III. NETWORK ASSUMPTIONS

According with the Gnutella 0.6 specifications [5], the network is assumed to be organized into two layers: the top layer is composed of a number of ultra-peers, while the bottom layer comprises a higher number of leaf-peers. Each leaf-peer is connected to a few ultra-peers, while each ultra-peer is connected to several other ultra-peers. An ultra-peer acts as a proxy to the top-layer network for the leaf-peers connected to it. A leaf-peer can submit a query only to its ultra-peers, which in turn forward the query to other ultra-peers using a TTL-limited flooding search. An ultra-peer forwards a query to a leaf-peer only if it believes the leaf-peer can answer it.

With this network structure we can distinguish three types of connections: leaf-peer to ultra-peer (LU), ultra-peer to ultra-peer (UU), ultra-peer to leaf-peer (UL). All three types follow a power law distribution for the number of connections (degree) of nodes [26][27]. For LU-type connections, the degree distribution can be modelled as a Zipf. For UU-type and UL-type connections, the degree distribution can be modelled better as a power law with exponential cutoff of the form $\mathrm{p(degree)} = \mathrm{degree}^{-\alpha}\mathrm{e}^{-\mathrm{degree/cutoff}}$, where degrees much greater than cutoff are very infrequent. A Zipf can also be used to model the distribution of file popularity and file size [28][29].

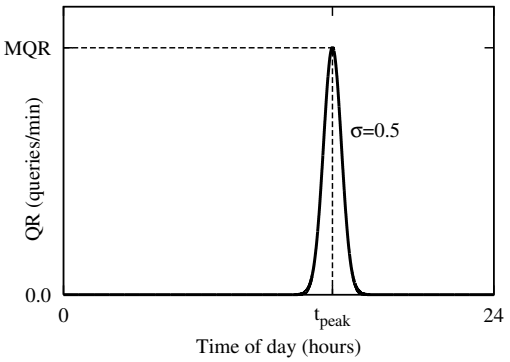

Figure 1. Query rate (QR) of a peer as a function of the time of day.

Regarding network activity, query submissions are modelled as Poisson processes: the inter-generation times of the queries submitted by any peer are independent and obey an exponential distribution with a given *query rate* (QR). According with [3], it is assumed that the QR of a peer reaches its *maximum query rate* (MQR) at a given time of the day, denoted t_{peak}, and distributes around the maximum following a Gaussian distribution (see Figure 1). The value of t_{peak} of a peer is a random real number uniformly distributed in the range $[0, 24[$ hours. This modelling aims at reproducing the behavior of a wide-area peer-to-peer network in which peers, being located in countries with different time zones, reach their maximum client-side activity at different hours.

IV. SLEEP-AND-WAKE ALGORITHM

To preserve the indexing and discovery infrastructure, ultra-peers of the Gnutella network are assumed to be always available, while leaf-peers can switch between *wake* and *sleep* mode over the time to reduce energy consumption. When a leaf-peer is in wake mode, it is available for download requests and works at normal power level. Conversely, a leaf-peer in sleep mode is unavailable, but it works at reduced power level, thus consuming a limited amount of energy.

Figure 2 illustrates how sleep-wake cycles of a leaf-pear evolve over the time. Initially, at the time denoted $W[0]$, the leaf-peer is in wake mode. At time $S[1]$, the first sleep-wake cycle begins, with the leaf-peer that goes in sleep mode for the first time. The first sleep period ends at time $W[1]$ when the leaf-peer returns in wake mode. The first sleep-wake cycle ends at time $S[2]$, when the second cycle begins. The duration of the i-th wake period, i.e. $S[i+1] - W[i]$, is greater than or equal to a constant WD. In particular, the duration will be equal to WD if at time $W[i] + WD$ the leaf-peer is not busy with any query processing or file transfer activity, and

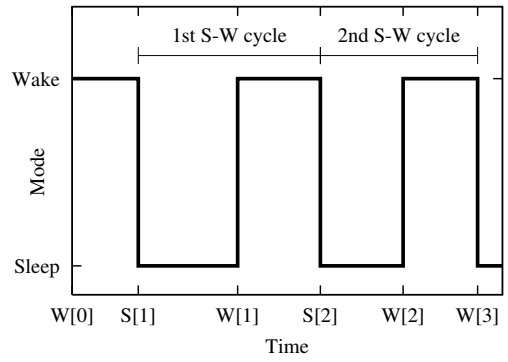

Figure 2. Sleep-wake cycles.

TABLE I
MEANING OF THE SYMBOLS USED IN FIGURES 3 AND 4.

Symbol	Meaning
t	Current time (logical clock)
WD	Min. duration of any wake period
$SD[i]$	Max. duration of the i-th sleep period
$W[i]$	Time when the i-th wake period begins
$S[i]$	Time when the i-th sleep period begins
t_{check}	Next time when p's activity will be checked
t_{wakeup}	Next time when p is scheduled to wake up
δ_{check}	Min. interval between two activity checks
δ_{sub}	Min. interval between query submission and sleep
δ_{hit}	Min. interval between query hit submission and sleep

is calculated that the next sleep duration is greater than zero. Otherwise, the beginning of the next sleep period is deferred and so the i-th wake period will be longer than WD. The duration of the i-th sleep period, i.e. $W[i]-S[i]$, is calculated by the leaf-peer at end of the $(i-1)$-th wake period based on the specific strategy adopted, as described in more detail in Section V.

Figure 3 illustrates the general *SleepAndWake* algorithm implementing the energy-saving scheme outlined above (see Table I for the notation used in the algorithm). The algorithm works as follows. Initially, some variables are initialized (*lines 1-5*), namely t, $W[0]$, t_{check}, t_{wakeup}, and i. Then, the following operations are cyclically performed.

If the current time is equal to t_{check} (*line 7*), it is further checked whether p is busy (*line 8*), by invoking the *isBusy* function described later in this section. If p is busy, the check for p's activity is deferred by a small amount of time δ_{check} (*line 9*). If p is not busy, the duration of the i-th sleep period $SD[i]$ is calculated (*line 11*) using one of the strategies discussed in the next section. If $SD[i]$ is greater than zero (*line 12*), $S[i]$ is set to the current time (*line 13*), the next wake up is scheduled at $S[i]+SD[i]$ (*line 14*), and then the leaf-peer goes in sleep mode (*line 15*). If instead $SD[i]$ is equal to zero (*line 16*), the check for p's activity is deferred by δ_{check} (*line 17*).

If the current time is equal to t_{wakeup} (*line 20*), $W[i]$ is set to the current time (*line 21*), the next activity check is scheduled at $W[i]+WD$ (*line 22*), the index of the next sleep-wake cycle is increased by one (*line 23*), and then the leaf-peer returns in wake mode (*line 24*).

Finally, the logical clock is increased by one (*line 26*). It is worth noticing that a sleep period can be interrupted earlier if a user interacts with the leaf-peer machine. In this case (for the sake of space not illustrated in Figure 3), t_{wakeup} is advanced to t, and the wakeup operations of *lines 21-24* are immediately executed.

The *isBusy* function is illustrated in Figure 4. The function returns that p is busy if at least one of the following conditions holds true: i) p submitted its last query no more than δ_{sub} seconds ago; ii) p sent its last query hit no more than δ_{hit} seconds ago; iii) p is currently uploading one or more files to other peers; iv) p is currently downloading one or more files from other peers.

```
// Executed by every leaf-peer p
SleepAndWake()
 1: t = 0; // clock reset
 2: W[0] = t;
 3: t_check = W[0] + WD;
 4: t_wakeup = −1;
 5: i = 1;
 6: while true do
 7:     if t == t_check then
 8:         if p.isBusy() then
 9:             t_check = t + δ_check;
10:         else
11:             Calculate SD[i];
12:             if SD[i] > 0 then
13:                 S[i] = t;
14:                 t_wakeup = S[i] + SD[i];
15:                 sleep();
16:             else
17:                 t_check = t + δ_check;
18:             end if
19:         end if
20:     else if t == t_wakeup then
21:         W[i] = t;
22:         t_check = W[i] + WD;
23:         i = i + 1;
24:         wakeup();
25:     end if
26:     t = t + 1; // clock increment
27: end while
```

Figure 3. General *SleepAndWake* algorithm executed by every leaf-peer.

```
// Returns true if p is busy
isBusy()
1: if t - p.lastQuerySubmissionTime() ≤ δ_sub or
      t - p.lastQueryHitSubmissionTime() ≤ δ_hit or
      p.isUploadingFile() or p.isDownloadingFile() then
2:     return true;
3: else
4:     return false;
5: end if
```

Figure 4. Function to check the activity status of a leaf-peer.

V. SLEEP STRATEGIES

Given the general *SleepAndWake* algorithm described in the previous section, it is possible to define different strategies to decide the duration of the next sleep period. In the remainder of the section we will introduce five strategies:

- VAR_HR: variable sleep duration depending on the *hit rate*;
- VAR_FS: variable sleep duration depending on the number of *files shared*;
- VAR_QR: variable sleep duration depending on the *query rate*;
- FIX_1WD: fixed sleep duration equal to WD;
- FIX_3WD: fixed sleep duration equal to $3WD$;

A. VAR_HR

We define *hit rate* of the i-th wake period of a leaf-peer p, denoted $HR[i]$, the number of query hits generated by p during the time interval $[W[i], t]$ divided by $t - W[i]$, where t is the ending time of the i-th wake period.

With the VAR_HR (VARiable with Hit Rate) strategy, the duration of the i-th sleep period of a leaf-peer p, denoted $SD[i]$, depends on $HR[i-1]$ as specified by the following equation:

$$SD[i] = \begin{cases} 0\ m & \textbf{if } HR[i-1] > 1\ h/m \\ 5\ m + \frac{10}{HR[i-1]} & \textbf{if } 0.1\ h/m < HR[i-1] \leq 1\ h/m \\ 120\ m & \textbf{otherwise} \end{cases}$$

In the equation above, $SD[i]$ is measured in minutes (here abbreviated to m), while HR is measured in hits per minute (abbreviated to h/m). Note that the equation above and those associated with the next two strategies include some constants whose values have been tuned experimentally.

Using the VAR_HR strategy, the leaf-peers with a high hit rate will not sleep at all or will sleep for a short amount of time, while those with a lower hit rate will sleep longer. In particular, $SD[i]$ is maximum for the so-called *free riders* (peers that do not share any file), because they do not generate any query hit.

B. VAR_FS

With VAR_FS (VARiable with Files Shared), the duration of the i-th sleep period of a leaf-peer p, $SD[i]$, depends on $FS[i-1]$, which represents the number of files shared by p at the end of the $(i-1)$-th wake period. In particular, $SD[i]$ is calculated as follows:

$$SD[i] = \begin{cases} 5\ m & \textbf{if } FS[i-1] > 100 \\ 15\ m + \frac{100\ m}{FS[i-1]} & \textbf{if } 1 \leq FS[i-1] \leq 100 \\ 120\ m & \textbf{otherwise} \end{cases}$$

Using this strategy, the leaf-peers with a high number of files will sleep for a short amount of time, while those with a lower number of files will sleep longer. Also in this case the free riders will sleep very long, because they do not share any file.

C. VAR_QR

Differently from the two strategies above, which link the sleep duration to some server-side characteristics of a leaf-peer (hit rate and number of files shared), the VAR_QR (VARiable with Query Rate) strategy links the sleep duration of a leaf-peer to its client-side behavior, namely, the *query rate* of the leaf-peer during the previous wake period.

We define *query rate* of the i-th wake period of a leaf-peer p, denoted $QR[i]$, the number of queries submitted by p during the time interval $[W[i], t]$ divided by $t - W[i]$. Given such definition, $SD[i]$ with VAR_QR is calculated through the following equation, where QR is measured in queries per minute (abbreviated to q/m):

$$SD[i] = \begin{cases} 0\ m & \textbf{if } QR[i-1] > 1\ q/m \\ 5\ m + \frac{10}{QR[i-1]} & \textbf{if } 0.1\ q/m < QR[i-1] \leq 1\ q/m \\ 120\ m & \textbf{otherwise} \end{cases}$$

D. FIX_1WD and FIX_3WD

FIX_1WD and FIX_3WD are two blind strategies with which all the sleeps have the same fixed duration. They are introduced mostly for comparison with the previous strategies. Specifically, with FIX_1WD (Fixed to WD) the sleep duration is equal to WD:

$$SD[i] = WD$$

while with FIX_3WD (Fixed to $3WD$), the sleep duration is equal to three times WD, that is:

$$SD[i] = 3WD$$

VI. PERFORMANCE EVALUATION

The goal of this section is to evaluate the amount of energy that can be saved in Gnutella by using the proposed *SleepAndWake* algorithm and the five strategies introduced above. To this end, as the main performance parameter, we will evaluate the *total energy consumption* (TEC) of the network over a period of observation. In addition, we will also evaluate the average *hit rate* (HR), i.e., the fraction of successful queries during the period of observation. The five strategies will be compared with a sixth strategy, referred to as NOSLEEP, in which all nodes are assumed to be always in wake mode.

A custom network simulator written in Java was used to carry out the evaluation. For the readers convenience, Table II reports the main simulation parameters. The simulator builds a Gnutella network composed by a *number of nodes* (NN)

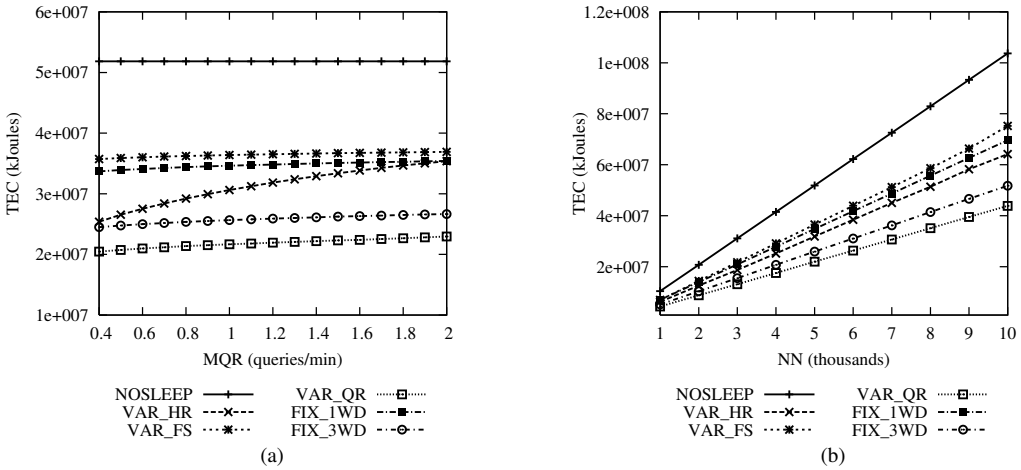

Figure 5. Total energy consumption: (a) NN=5000 and MQR variable; (b) NN variable and MQR=1.2.

TABLE II
SIMULATION PARAMETERS.

Parameter	Description	Values
NN	Number of nodes	1000-10000
MQR	Max. query rate (queries/min.)	0.4-2.0
SRD	Simulation run duration	24 h
$WMPC$	Wake-mode power consumption of every peer [32]	120 W
$SMPC$	Sleep-mode power consumption of every peer [32]	5 W
$WSTT$	Wake-to-sleep transition time [33]	9 s
$SWTT$	Sleep-to-wake transition time [33]	4 s
TTL	Time-to-live [34]	3
MTT	Message transfer time [35]	20 ms
QPT	Query processing time [36]	2 ms
$QHPT$	Query hit processing time [36]	1 ms
WP	Min. duration of a wake period	20 min.
δ_{check}	Min. interval between two activity checks	1 min.
δ_{sub}	Min. interval between query submission and sleep	10 s
δ_{hit}	Min. interval between hit submission and sleep	10 s

ranging from 1000 to 10000, the 15% of which configured to play the role of ultra-peers, according with [30]. Regarding the three types of connections introduced in Section III, LU-type degrees follow a Zipf with alpha equal to 1.4 [26], while UU-type and UL-type degrees follow a power law with exponential cutoff with alpha equal to 2.2 and cutoff equal to 32 [27][31]. According with [28] and [29], the alpha parameter of the Zipf distribution of file popularity and file sizes is equal to 1.2, while according with [29], the 15% of peers do not share any file. It is also assumed that the number of distinct files is three times the number of nodes. Queries are submitted to the network by peers as explained in Section III, with a *maximum query rate* (MQR) that ranges between 0.4 and 2.0 queries per minute.

A. Total Energy Consumption

Figure 5a shows the TEC of a network with $NN = 5000$ and MQR ranging from 0.4 to 2.0, after 24 hours of simulated activity using the five strategies introduced in Section V and the NOSLEEP strategy introduced earlier in this section. As shown in the figure, NOSLEEP is the most energy-consuming strategy because it keeps all nodes always in wake mode. Therefore, the TEC of NOSLEEP is taken as the reference value for estimating the amount of energy that can be saved using the sleep-and-wake strategies proposed in this paper.

Based on the simulation results, VAR_QR is the strategy ensuring the highest energy saving potential, since its TEC is on average 42% of that obtained with the NOSLEEP strategy. The second best strategy in terms of energy saving is FIX_3WD, whose TEC is on average 50% of that obtained with NOSLEEP. Third strategy resulted VAR_HR (61%), followed by FIX_1WD (67%) and VAR_FS (70%). All the strategies (but NOSLEEP) increase their TEC as MQR increases. This depends on the fact that high MQR values mean that peers submit more queries, which increases the possibility that a sleep period will be deferred due to the consequent client-side or server-side activity. This is particularly evident in the case of VAR_HR, because the hit rate grows proportionally with the number of queries submitted to the network, thus reducing the overall sleep time and increasing the TEC value.

Figure 5b shows the TEC of the network with NN ranging from 1000 to 10000 and $MQR = 1.2$. As expected, the simulation results show that the TEC increases linearly with the network size. Therefore, the absolute amount of energy that can be saved using the proposed algorithm and strategies increases significantly as the network grows in size.

B. Hit Rate

Figure 6a shows the HR of a network with $NN = 5000$ and MQR ranging from 0.4 to 2.0, using the five proposed strategies and NOSLEEP. In this case NOSLEEP ensures the highest hit rate (about 87%), because all peers are always in wake mode and therefore all files are always available. Note that even if all peers are always in wake mode, NOSLEEP does not reach 100% hit rate because the TTL used guarantees only 85%-90% of network coverage.

While the lowest TEC value is ensured by VAR_QR, the

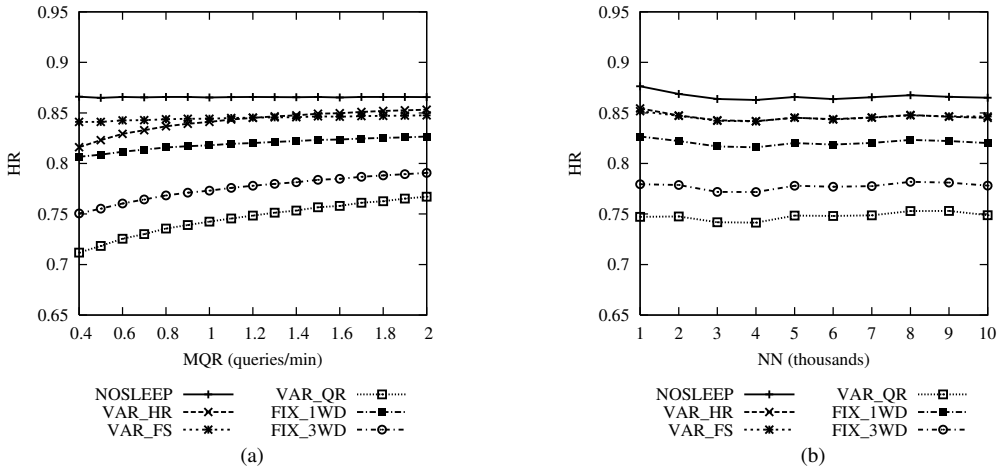

Figure 6. Hit rate: (a) NN=5000 and MQR variable; (b) NN variable and MQR=1.2.

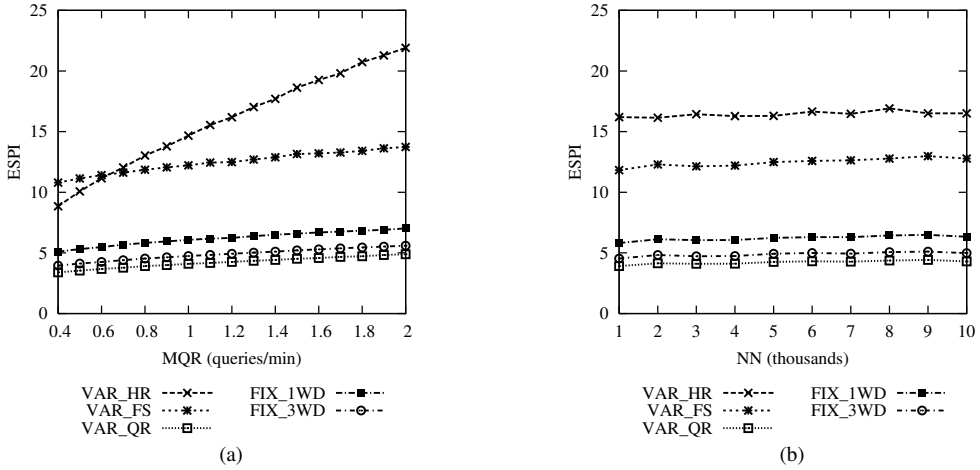

Figure 7. Energy-search performance index: (a) NN=5000 and MQR variable; (b) NN variable and MQR=1.2.

highest HR value is obtained with VAR_FS and VAR_HR: the former provides the best hit rate with $MQR \leq 1.2$, the latter ensures the best result with $MQR > 1.2$. All the strategies (but NOSLEEP) increase their HR as MQR increases. As for the case of TEC, high MQR values increase the possibility that a sleep period will be deferred, which increases the overall time passed in wake mode by the peers and consequently the possibility that a file is available when it is searched.

Figure 6b shows the HR with NN ranging from 1000 to 10000 and $MQR = 1.2$. Also in this case, the result is in line with expectation, i.e., the hit rate does not depend on the number of nodes in the network, with only some minor differences of HR values due to stochastic variations while the network is created by the simulator.

C. Energy-Search Performance Index

Let $TEC(\text{NOSLEEP})$ and $HR(\text{NOSLEEP})$ be respectively the total energy consumption and hit rate obtained with the NOSLEEP strategy in a given network scenario, and let $TEC(\text{X})$ and $HR(\text{X})$ be respectively the total energy consumption and hit rate obtained with a strategy X in the same scenario, where X is one of the five strategies introduced in Section V.

We define *energy-search performance index* ($ESPI$) of a strategy X in a given network scenario the following number:

$$ESPI(\text{X}) = \frac{\frac{TEC(\text{NOSLEEP}) - TEC(\text{X})}{TEC(\text{NOSLEEP})}}{\frac{HR(\text{NOSLEEP}) - HR(\text{X})}{HR(\text{NOSLEEP})}}$$

The $ESPI$ is used in our study as an aggregate performance indicator to provide an overall evaluation of the proposed strategies taking into account both energy efficiency and search quality.

Given the definition above, it is desirable that the $ESPI$ of a given strategy is greater than 1 for each combination of MQR and NN: lower values would be indicative of a hit ratio degradation higher than the benefit obtained in terms of energy saving. Figures 7a and 7b shows that the $ESPI$ values for all the proposed strategies are actually greater than 1 for each combination of MQR and NN.

In particular, Figure 7a shows that the best strategies, based on their $ESPI$ values, are VAR_FS and VAR_HR: the former with $MQR < 0.8$, the latter with $MQR \geq 0.8$. Once again, the result is independent from network size, as shown in Figure 7b.

VII. CONCLUSIONS

Reducing the energy consumption of large-scale distributed systems is an challenging task, as it involves design and optimization of energy-aware algorithms, architectural models, and applications. This is particularly true in peer-to-peer file sharing networks, given the large sets of nodes participating to such systems, and the need of reducing energy consumption without compromising the quality of service delivered to the users [3].

In this paper we focused on evaluating how the sleep-and-wake energy-saving approach can be used to reduce energy consumption in Gnutella, one of the most popular peer-to-peer file sharing networks. In order to save energy, we introduced a general sleep-and-wake algorithm that allows leaf-peers of the Gnutella network cyclically switch between wake and sleep mode, where the time passed in sleep mode is autonomously decided by each leaf-peer.

We defined different strategies that a leaf-peer may employ to decide the duration of its sleep periods. Simulation results have proven the effectiveness of the sleep-and-wake approach in different network scenarios, as well as the performance of the proposed sleep strategies in terms of energy efficiency and search quality.

REFERENCES

[1] H. Schulze, and 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.

[2] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, "Identifying and discriminating between web and peer-to-peer traffic in the network core," WWW 2007.

[3] P. Trunfio, "A two-layer model for improving the energy efficiency of file sharing peer-to-peer networks", Concurrency and Computation: Practice and Experience, 2014. DOI: 10.1002/cpe.3213.

[4] A. Malatras, F. Peng, B. Hirsbrunner, "Energy-efficient peer-to-peer networking and overlays," In: M. S. Obaidat, A. Anpalagan, and I. Woungang (Eds.), Handbook of Green Information and Communication Systems, Elsevier, 513-540, 2013.

[5] Gnutella 0.6, RFC Gnutella 0.6, June 2002, http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[6] G. Anastasi, I. Giannetti, A. Passarella, "A BitTorrent proxy for Green Internet file sharing: Design and experimental evaluation," Computer Communications 33(7):794-802, 2010.

[7] B. Cohen, "Incentives Build Robustness in BitTorrent," Workshop on Economics in Peer-to-Peer Systems 2003.

[8] P. Purushothaman, M. Navada, R. Subramaniyan, C. Reardon, A.D. George, "Power-Proxying on the NIC: A Case Study with the Gnutella File-Sharing Protocol," LCN 2006.

[9] T. Enokido, A. Aikebaier, M. Takizawa, "A Model for Reducing Power Consumption in Peer-to-Peer Systems," IEEE Systems Journal 4(2):221-229, 2010.

[10] T. Enokido, K. Suzuki, A. Aikebaier, M. Takizawa, "Laxity Based Algorithm for Reducing Power Consumption in Distributed Systems," CISIS 2010.

[11] I. Kelenyi, J. K. Nurminen, "Optimizing Energy Consumption of Mobile Nodes in Heterogeneous Kademlia-based Distributed Hash Tables," NGMAST 2008.

[12] P. Maymounkov, and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," IPTPS 2002.

[13] M. S. Joseph, M. Kumar, H. Shen, S. Das, "Energy Efficient Data Retrieval and Caching in Mobile Peer-to-Peer Networks," PERCOMW 2005.

[14] K. Park, P. Valduriez, "Energy Efficient Data Access in Mobile P2P Networks," IEEE Trans. Knowl. Data Eng. 23(11):1619-1634, 2011.

[15] Y.-C. Tung, K. C.-J. Lin, "Location-Assisted Energy-Efficient Content Search for Mobile Peer-to-Peer Networks," 7th International Workshop on Mobile Peer-to-Peer Computing, 2011.

[16] A. Ka-Ho Leung, Y.-K. 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.

[17] J.-S. Han, J.-W. Song, T.-H. Kim, S.-B. Yang, "Double-layered Mobile P2P Systems Using Energy-Efficient Routing Schemes," ATNAC 2008.

[18] G. Lefebvre, M. J. Feeley, "Energy Efficient Peer-to-Peer Storage," Tech. Report (TR-2003-17), Dept. of Computer Science, University of British Columbia, 2003.

[19] J. Blackburn, K. Christensen, "A Simulation Study of a New Green BitTorrent," ICC 2009.

[20] Y.-J. Lee, J.-H. Jeong, H.-Y. Kim, C.H. Lee, "Energy-Saving Set-Top Box Enhancement in BitTorrent Networks," NOMS 2010.

[21] S. Gurun, P. Nagpurkar, B. Y. Zhao, "Energy Consumption and Conservation in Mobile Peer-to-Peer Systems," MobiShare 2006.

[22] A. Sucevic, L. L. H. Andrew, T. T. T. Nguyen, "Powering Down for Energy Efficient Peer-to-Peer File Distribution," ACM GreenMetrics 2009.

[23] G. Jourjon, T. Rakotoarivelo, M. Ott, "Models for an Energy-Efficient P2P Delivery Service," PDP 2010.

[24] L. L. H. Andrew, A. Sucevic, T. T. T. Nguyen, "Balancing Peer and Server Energy Consumption in Large Peer-to-Peer File Distribution Systems," GreenCom 2011.

[25] H. Hlavacs, K. A. Hummel, R. Weidlich, A. M. Houyou, H. de Meer, "Modeling Energy Efficiency in Distributed Home Environments," International Journal of Communication Networks and Distributed Systems, 4(2):161-182, 2010.

[26] K. Wehrle, R. Steinmetz (Eds.), "Peer-to-Peer Systems and Application", LNCS 3485, Springer, 2005.

[27] S. K. Shaw, "Topology Formation and Replication Strategies for Gnutella Network", M.Tech. Thesis, Indian Institute of Technology, Kharagpur, 2008.

[28] H. Jin, H. Chen, "SemreX: Efficient search in a semantic overlay for literature retrieval", Future Generation Computer Systems 24(6): 475-488, 2008.

[29] D. Stutzbach, S. Zhao, R. Rejaie, "Characterizing files in the modern Gnutella network", Multimedia Syst. 13(1): 35-50, 2007.

[30] D. Stutzbach and R. Rejaie, "Capturing Accurate Snapshots of the Gnutella Network," INFOCOM 2005.

[31] Y. Wang, X. Yun, Y. Li, "Analyzing the Characteristics of Gnutella Overlays", ITNG 2007: 1095-1100.

[32] G. Knezek, R. R. Christensen, T. Tyler-Wood, O. Lim, W. E. Neaville, "Going green with IT: a study of energy consumption by Home and School Information Technology Systems in the College of Information at the University of North Texas", iConference 2010.

[33] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, R. Gupta, "Somniloquy: augmenting network interfaces to reduce PC energy usage", NSDI'09: 365-380.

[34] S. K. Shaw, J. Chandra, N. Ganguly. "HPC5: An Efficient Topology Generation Mechanism for Gnutella Networks". ICDCN 2009: 114-126.

[35] S. Saroiu, P. K. Gummadi , S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", MMCN 2002.

[36] W. Müller, M. Eisenhardt, A. Henrich, "Scalable summary based retrieval in P2P networks", CIKM 2005: 586-593