Large-scale Distributed Systems and Energy efficiency: a Holistic View, Edited by Jean-Marc Pierson.

ISBN 0-471-XXXXX-X

Chapter 9

Energy Efficiency in P2P Systems and Applications

Simone Brienza[1], Sena Efsun Cebeci[2], Seyed-Saeid Masoumzadeh[3], Helmut Hlavacs[3], Öznur Özkasap[2], Giuseppe Anastasi[1]

[1] Dept. of Information Engineering, University of Pisa, Italy
E-mail: simone.brienza@for.unipi.it, giuseppe.anastasi@iet.unipi.it

[2] Department of Computer Engineering, Koç University, Istanbul, Turkey
E-mail: senacebeci@ku.edu.tr, oozkasap@ku.edu.tr

[3] Research Group Entertainment Computing, University of Vienna, Austria
E-mail: masoumzadeh@gmail.com, helmut.hlavacs@univie.ac.at

P2P systems and applications have achieved increasing popularity in the last years due to the characteristics of dynamicity and scalability of the peer-to-peer paradigm. Currently, P2P applications – especially file sharing and file distribution applications – generate a remarkable portion of the overall Internet traffic. However, many common P2P protocols do not consider the energy problem. Frequently, hosts are requested to stay on and connected to the network for long times. Therefore, they are very energy-consuming. In this chapter, we present a general taxonomy to classify possible approaches to the energy problem in P2P systems and applications. Then, we survey the main solutions available in the literature, focusing on two relevant classes of P2P protocols, namely file sharing/distribution protocols (e.g., BitTorrent and Gnutella) and epidemic P2P protocols.

I. Introduction

Peer-to-Peer (P2P) systems have gained more and more importance in recent years. P2P file sharing applications are among the most popular Internet applications and account for a large fraction of the overall Internet traffic [1]. According to an experimental study carried out in 2009 [2], the percentage of the Internet traffic originating fromP2P file sharing ranges from 40% to 73%, depending on the considered geographic area. Also many distributed services such as reliable multicasting, aggregate computation, frequent items discovery, overlay topology construction, and failure detection rely on P2P systems. Originally, P2P systems have been conceived without considering the energy problem. For instance, BitTorrent [3] – which is the most popular protocol for P2P file sharing – has no internal mechanism for energy efficiency. The main motivation for this design choice is that, in P2P systems, energy consumption is distributed among a large number of peers, while in traditional client-server architectures energy is mainly consumed at data centers. Hence, in P2P systems energy costs are not supported by a single organization, instead they are shared by a large number of users. As the energy costs of P2P systems and applications keep increasing with their popularity and share of the overall Internet traffic – and awareness about energy and environmental issues increases – developing energy efficient solutions for P2P systems has attracted more and more attention in recent years.

A number of solutions have been proposed in the literature. Since energy consumption is distributed among several peers taking part in the application, providing energy efficient solutions for P2P systems is generally more challenging than for traditional client-server scenarios. This issue is pointed out in [4] where it is shown that energy adaptation in P2P

environments is more complex than that in a traditional client-server environment. In this chapter we classify the main approaches to energy efficiency in P2P systems and applications, by introducing a general taxonomy. Then, we survey the main solutions proposed in the literature, with special focus on two main areas, namely file sharing/distribution and epidemic protocols.

The rest of this chapter is organized as follows. Section II discusses the main approaches to energy efficiency in P2P systems and applications and introduces a general taxonomy for them. Section III addresses energy efficiency in P2P file sharing applications, while Section IV focuses on energy efficiency in P2P epidemic protocols. Section V draws some conclusions.

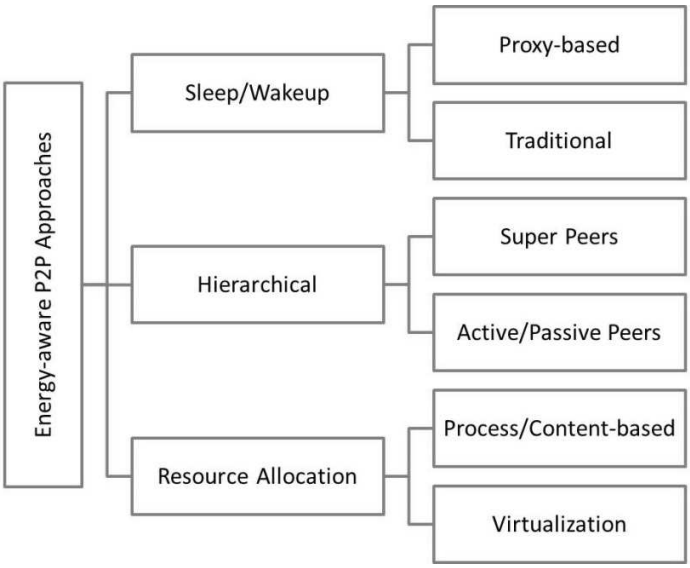

Figure I. Classification of energy-aware P2P approaches

II. General Approaches to Energy Efficiency

In this section, we provide a classification of energy-aware P2P approaches into three main classes, based on the techniques they use, namely *sleep/wakeup, hierarchical* and *resource allocation.* The taxonomy is illustrated in Figure I.

A. Sleep/Wakeup Approaches

Sleep/Wakeup is a commonly used energy-efficiency technique in P2P systems, which is based on the idea of switching off some of the unused devices in the system and waking them up when necessary. Since availability has utmost importance in P2P systems, sleep/wakeup modes should be designed smartly. Furthermore, due to their dynamic nature, P2P systems require even smarter techniques. Sleep/wakeup approaches can be further divided into two sub-classes, namely *proxy-based* and *traditional*, depending on whether, or not, they use proxy sub-system. The two alternative approaches are discussed below.

Proxy-based Sleep/Wakeup

In proxy-based solutions, peers either completely hand over their tasks to a proxy or use the proxy only for the purpose of waking up. Proxy-based solutions for Gnutella P2P file sharing [5] have been proposed in [6] and [7]. In both cases the idea consists in using a P2P proxy subsystem – either embedded in the Network Interface Card (NIC) at the host or in another device such as a LAN switch – that takes over the control when the host goes into a low-power mode.

EE-BitTorrent [8] [9] also leverages a proxy-based architecture for improving energy efficiency in BitTorrent [3]. The authors point out that energy waste could be avoided by

switching off some peers and delegating the file download to their associated proxy. The proxy can be either a dedicated computer, or a machine that has to be continuously powered on for providing other network services.

Download Sharing [10] also relies on proxying to reduce the energy consumption of P2P file sharing when dealing with unpopular files, i.e., files for which very few copies are available on the overlay. Since the number of seeders is very low, P2P file sharing tends to degenerate in a client-server file transfer and the download time (and energy consumption) increases significantly. In such conditions it would be more convenient to combine the upload capacity of seeders and avoid semantically redundant download flows. To this end, in Download Sharing a number of *active leechers* act as proxies and manage the download task on behalf of passive leechers, whose power mode is changed into sleep mode to ensure energy savings. Active leechers are selected among peers with the fastest links.

Proxy-based approaches are also used for P2P content sharing in wireless mobile systems [11][12][13][14][15][16]. All the previous proposals will be analyzed in detail in Section III.

Traditional Sleep/Wakeup

In traditional sleep/wakeup approaches, peers do not use a proxy subsystem and the wakeup operation of a peer in sleep mode is done either by the peer itself or in cooperation with the other peers. An example study in this group aims at building an energy efficient P2P storage system [17]. It argues that idle network resources are abundant in P2P systems and sleep mode can be utilized to save energy. Before entering

the sleep mode, peers compute a future wakeup time. When this period of time passed, they wake up if there is no failure.

In the context of file sharing *Green BitTorrent* [18] takes a traditional sleep/wakeup approach. It allows peers to sleep without being dropped from peer lists. By modifying the behavior of the BitTorrent client to flag a disconnecting peer as asleep, instead of removing its entry from the peer list, it is possible to maintain knowledge of sleeping peers throughout the swarm. If the resources of the sleeping peer are later needed, a peer can wake it up and restore the TCP connection. Green BitTorrent relies on the wake-on-LAN (WoL) technology [19]. A similar approach is also used in [20], where a set-top-box prototype for energy efficiency in BitTorrent is proposed. Like Green BitTorrent, it defines new peer states, including energy saving mode, uses wake-on-LAN technology and relies on *hibernation* and *wakeup* messages.

An adaptive BitTorrent mechanism is proposed in [21], which relies on a *seed pool* method. The elements of the pool are servers with diverse upload capacities and power consumption rates. Sleep and active states are defined for all peers in the swarm. Peers in active state perform actions to serve the file download requests from other peers. For energy efficiency, seeders with high upload rates help peers with low upload rate to finish the download process quickly.

A model developed to reduce energy consumption in datacenters is proposed in [22]. For this purpose, it uses a distributed algorithm, deployed over a P2P network of Set-Top-Boxes (STBs). STBs are similar to small computers with hard-drives, CPU, and even GPU processors. In this study, the aim is to put unused STBs in standby mode without degrading the general Service Level Agreement (SLA). STBs are turned on only when

clients request the service. The study discusses the potential advantages of turning STBs off and evaluations show energy consumption improvements.

B. Hierarchical Approaches

Hierarchical approaches introduce a structure in the overlay network. Not all peers are equal but some peers have a special role in the overlay. Hierarchical approaches can be further divided in two different classes, depending on whether they use *super peers*, or not. Super peers are special peers that are advantageous in terms of energy efficiency and are preferable when compared to other peers. They are selected within the overlay to carry out some specific tasks. In hierarchical approaches that do not rely on super peers, referred to as hierarchical approaches with *active/passive peers* in our taxonomy, peers can be either active or passive. The aim of these approaches is to reduce the number of active peers in the system and, hence, the network overhead and resource usage.

Hierarchical Approaches based on Super Peers

An example of hierarchical solution based on super peers can be found in [23], where the authors propose an energy efficient routing approach for *double-layered* mobile P2P systems aimed to improve the system lifetime. In a double layered P2P system there are two types of peers, namely *super peers* and *sub-peers*, and the search process is done mainly through super peers. Each super peer in the proposed system maintains a route table and a file routing table to keep the information regarding the routes and file lists, respectively.

Another example of solution based on super peers is presented in [24] and it develops a system where there exist multiple server computers each of which holds a file. A client

computer sends a file transfer request to a load balancer (i.e., a logical process). Then, the load balancer selects one server computer s in the set S. There are two main conditions that must be considered when selecting the server computer. Firstly, it should satisfy the deadline constraint for the file transmission. Secondly, the power consumption of the selected server computer is also taken into account.

An energy efficient P2P agreement protocol is presented in [25], which aims to reduce redundant transmissions without compromising reliability. For this purpose, it uses a mechanism, named TBMPR (*Trustworthiness-Based MultiPoint Relaying*), which determines the trustworthiness of each peer by the number of successful transactions. Each peer sends a message to neighboring peers and only trustworthy neighbor peers forward that message.

Hierarchical Approaches based on Active/Passive Peers

Hierarchical approaches based on active/passive peers have been considered for energy efficiency in epidemic protocols. In [26] and [27] the authors investigate the power-awareness features of *flat* and *hierarchical* epidemic protocols. These studies propose a dominating-set based and power-aware hierarchical epidemic approach that eliminates a significant number of peers from gossiping. In contrast to previous works on hierarchical epidemics, the dominating set idea is used to construct a hierarchy, and to choose peers performing gossip operation. In this adaptive approach, only a subset of peers is active in gossiping, forming an overlay consisting of dominating set peers, so that the other peers can switch to idle, or passive state.

The hierarchical approach based on active/passive peers has been considered also for content sharing. The Smart Mobile Cloud (SMC) [28] is an energy efficient cloud-based

P2P content sharing solution for mobile devices. SMC uses a strategy for grouping mobile nodes (i.e., *helper nodes* act as active entities that manage the file download process, and *seeker nodes* request file downloads as passive entities) close to each other based on their location. Serving file download requests by using resources of the nodes in the same group would then facilitate energy efficiency.

Another energy adaptive technique for P2P file sharing in mobile devices is proposed in [29], where the idea is to gather mobile users according to their energy budget and to restrict their transmission capacity in order to achieve energy savings. In this technique, peers are grouped into *energy sufficient* and *energy constrained* peers. The download process differs from traditional BitTorrent protocol in the following aspects. When a file is requested, the tracker provides two neighbor lists to a peer. The first list contains information about peers belonging to the same energy group as the requester peer, while the second list contains information about energy-sufficient peers.

C. Resource Allocation

Resource allocation techniques have a significant effect on reducing energy consumption in P2P networks. These techniques are applied via various resource allocation mechanisms such as *process/content-based* or *virtualization*. In the *process/content-based* approaches, the idea is to allocate system resources in an energy efficient way. While some studies aim at reducing energy consumption of peers via efficient *process* allocation algorithms, others consider *content* file types as resources (e.g., polluted and unpolluted content) and conduct energy efficiency analysis for various system conditions. On the other hand, in *virtualization* approaches, system resources are managed by virtual machines to perform tasks in an energy efficient manner.

Process/Content-based Resource Allocation Approaches

In the category of content-based resource allocation, a model that considers content pollution for energy consumption is proposed in [30]. It is shown that deleting polluted file copies from stable systems and flash crowd systems yields energy savings. In the experimental analysis the effects of leave rate of clean copies and delete rates of polluted copies on energy consumption are explored. The results show that increasing the delete and leave rate has a positive impact on energy savings. On the other hand, in the flash crowd case, results demonstrate that an increase in the leave rate of clean copies reduces energy consumption.

In the category of process-based resource allocation, computing the amount of electric power consumed to perform processes in each computer is addressed in [31]. It uses a macro level model to show the relation between the amount of computation and the total power consumption of peer computers in order to perform Web types of application processes. In addition, the authors discuss the *laxity* concept for allocating a process to a computer so that the deadline constraint is satisfied and the total power consumption is reduced. In this system, there is a deadline constraint to allocate processes to computers. A deadline constraint on a process is issued by the application and the process has to be allocated to a computer so that it can terminate by the deadline. In laxity-based algorithm, the number of processes that do not satisfy the deadline constraint is reduced and evaluations show that the laxity-based process allocation algorithm is more efficient in reducing total power consumption when compared to the traditional round-robin algorithm.

Virtualization Resource Allocation Approaches

Some studies propose virtualization approach to conserve energy in P2P systems [32][33][34]. They are based on the idea of energy-efficient resource sharing among home networks interconnected via a P2P overlay, through virtualization of tasks. Virtualization of tasks means moving the virtual machine responsible for a particular task from one Virtual Home Environment (VHE) to another VHE. Shifting load to a small number of computers provides that the relieved computers can be hibernated to save energy.

A task virtualization prototype for unpopular files has been developed in [10], where a virtual machine (VM) that remotely takes care of managing the resources for file download is used. By using the VM the download tasks are migrated to other machines. However, this approach has drawbacks in terms of privacy and security. The authors have developed scenarios to investigate the resource utilizations of CPU, memory and NIC, and the results show that resource utilizations grow linearly while the number of VMs involved in the system increases. Furthermore, the effects of load, upload bandwidth and fairness on the energy consumption were demonstrated.

III. Energy Efficiency in File Sharing Applications

In this section, we focus on file sharing/distribution applications, which are the most popular P2P applications and, according to a number of studies [1] [2], account for a large fraction of the overall Internet traffic. In the next sub-section we compare, in terms of energy efficiency, the two main approaches to file sharing, i.e., client-server and P2P. Then, we focus on P2P systems and present some techniques for optimizing the energy

efficiency in P2P file sharing. Finally, we discuss energy efficiency in BitTorrent and in other popular P2P file-sharing protocols.

A. Client-Server vs. P2P File Sharing

A major difference between client-server networks and P2P ones is the role of each node. In a typical client-server network, each node can be either a *client* or a *server*, but not both. On the contrary, in a P2P network the role of each node is the same: it is both a client and a server at the same time. Each node is able to obtain and provide services from and to others simultaneously. With this in mind, in a P2P file-sharing system the file is not offered at a single server location, but is offered by a multitude of peers. Therefore, the load can be distributed among those peers. The increase in the number of clients and, consequently, the increase in file requests in a client-server approach can lead to performance degradations due to server overloading. The situation in a P2P system is less severe, since more clients not only mean more requests, but also higher service capacity and better scalability [35]. On the other hand, the robustness of a P2P file sharing system is also much stronger than that of client–server one, as each peer can be a file server and, consequently, failures of peers have less impact on the system than the failures of servers in client-server file-sharing systems.

To compare the energy consumption of P2P file-sharing systems to client-server systems, it is important to have an overall perspective of their architectures. Nedesvschi et al. [36] addressed some important factors contributing to energy consumption of both architectures and investigated which architecture is more energy efficient. In the following, we take these factors into consideration. In a P2P file sharing system peers typically spend a high portion of the day fully powered-up, while their average utilization

is typically very low. This factor plays a significant role in energy consumption of P2P systems. Apart from this, a P2P file-sharing system makes much heavier use of the network than a client-server system. It arises from generating overhead traffic for peer discovery and searching, as well as for membership maintenance. The average path length between peers can be significantly longer than the average path length between a client and the central server. On the other side, in a client-server file sharing architecture, the power consumed by cooling in data centers is the most important factor that reduces energy efficiency, while a P2P architecture is exempt from this cost due to its geographic distribution and the fact that peers are mostly home PCs. Another factor is the high baseline energy consumption of computers, i.e., the amount of energy that a computer requires to be powered up. The baseline energy consumption is typically much higher than the additional energy consumed when the utilization increases. The difference between P2P and client-server file-sharing systems lies in the fact that a P2P service is only responsible for the resultant increase in the peer's power consumption and is not responsible for baseline energy consumption, while in client-server architecture, the data center is responsible for the entire power consumption. The authors of [36] considered all aforementioned factors in the energy consumption modeling and compared P2P and client-server architectures to each other. They concluded that from the point of view of end systems, P2P is likely to always win. However, this conclusion strongly depends on the assumption that baseline energy consumption at peers comes for free.

B. Energy Efficiency in P2P File Sharing

In this section, we discuss a couple of theoretical approaches that have been proposed for improving the energy efficiency of P2P file-sharing systems [37] [38]. Sucevic et al. [37]

have proposed scheduling in order to power down peers. The authors propose different strategies for distributing the file to peers (i.e., *descending, random, ascending* upload-capacity order) and compare them with simultaneous and sequential approaches. In the ascending upload-capacity-order strategy, the scheduling mechanism involves peers finishing their download in an increasing order of their upload capacity and then going offline as soon as their download has been completed. This mechanism guarantees that peers with high request serving capabilities remain online for a longer time. The authors present a mathematical model and propose an optimal strategy for a small network. Further, their results show that the optimal strategies quickly become too complex as the number of peers grows. Based on the same concept, Andrew et al. [38] have proposed a scheduling strategy to turn peers off and on so as to minimize the total energy consumption. They have shown that the optimal case is achieved when peers are on only when they are downloading with the highest rate, while a small subset of energy-efficient peers might be needed to be on for the whole time of the transfer. The proposed strategy is centralized as the authors considered a central algorithm for scheduling, in contrast to the distributed nature of P2P systems.

C. Energy Efficiency in BitTorrent

In this section, we survey the main solutions proposed for increasing the energy efficiency of BitTorrent that is the most popular platform for P2P file sharing. In order to facilitate the reader's understanding, we preliminarily provide a brief description of the BitTorrent protocol (details can be found in [3][39]).

BitTorrent implements an unstructured overlay network customized for file sharing. Nodes in the overlay are called *peers*, and the set of peers involved in the distribution of a

file is referred to as *torrent* (or *swarm*). Each peer downloads the desired file, in *chunks*, from a multitude of other peers. While downloading missing chunks, peers upload to other peers in the same torrent the chunks they have already obtained. A peer can be either a *leecher*, if it does not have a complete copy of the file, or a *seeder*, if it has already achieved a complete copy of the file (seeders only upload chunks to other peers). For each torrent there is a *tracker*, i.e., a node that constantly tracks which peers are involved in the torrent. A peer that wants to join a torrent must register with the tracker and, then, it must periodically inform the tracker that it is still in the torrent. Figure II shows the different phases in the download process. As a preliminary step, a peer contacts the tracker and receives a random list of peers belonging to the torrent. It can thus open a TCP connection to a number of them, and start exchanging chunks of the file. At any given time, the peer will be in contact with a set of other peers, called *neighbors*. The set of neighbors changes dynamically as peers join and leave the torrent. In addition, each peer preferentially selects, for downloading chunks, those peers from which it can achieve the highest download rate (see below). Finally, every 30 seconds the peer randomly selects a new peer from the list, as a way to discover new neighbors and allow new peers in the torrent to start-up. At any given time, the peer has a subset of chunks composing the file. To figure out where missing chunks can be downloaded from, it periodically asks each of its neighbors for the list of chunks they have. Then, it uses the *Rarest First* strategy to decide the missing chunks to be requested first, i.e., it gives priority to chunks that are less spread in the torrent. Finally, to decide which requests from other peers have to be served, the peer uses the *Tit-for-Tat* (*TFT*) strategy, i.e., it gives priority to the peers from which it is downloading data at the highest rate.

Specifically, it measures the data rate it is achieving from each of its current neighbors and, then, selects the four neighbors with the highest data rate.

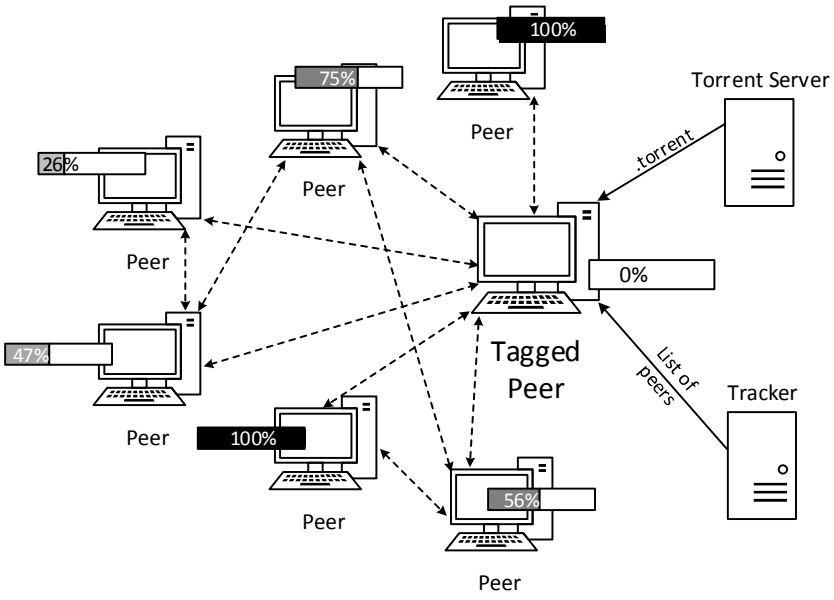

Figure II. File Distribution Process.

The BitTorrent protocol described above is not energy efficient. Peers have to stay active and connected to the overlay network for all the time required to download a file, which may take even several hours. Turning off peers is not a viable solution, for the following reasons. If the peer is a leecher (i.e., it is still downloading chunks of the file), powering it off does not provide any benefit in terms of energy savings, as the download process immediately stops. If the peer is a seeder, powering it off is apparently a very good idea, as it has already obtained a complete copy of the file. However, if the number of seeders decreases, the average time for downloading a complete copy of the file becomes larger and larger, resulting in a corresponding increase in the average energy consumption of peers. Hence, powering off a seeder may also result in an increase in the overall system energy consumption. Despite the previous remarks, a number of solutions have been

proposed to increase the energy efficiency of BitTorrent. Below we will discuss the most important of them.

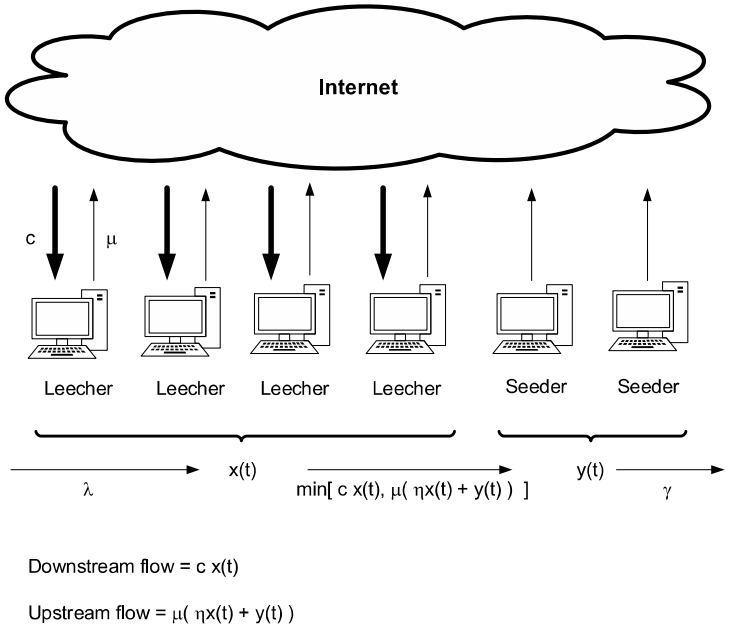

Figure III. Fluid model consisting of uploads and downloads.

Due to the protocol popularity, the general problem of BitTorrent performance optimization has been thoroughly investigated – also using analytical techniques – and a number of solutions have been proposed. Most of these studies, however, are aimed at minimizing the *average download time*[1], i.e. the average time experienced by a peer to download a complete copy of the file [40][41][42]. Minimizing the average download time is a very important goal, especially from the user perspective. However, this is typically achieved enforcing all (or a large number of) seeders to remain active all the time to continuously upload chunks, i.e. at the cost of an increased system energy consumption [38]. Hence, such strategies are inefficient from the energy consumption

[1] Another metric that is often used is the *last download time*, i.e., the time taken by the last peer to get a complete copy of the file.

point of view. To make BitTorrent energy efficient, the *system energy consumption* (i.e., the total energy consumed by all the peers to achieve a complete copy of the file) should be minimized, instead of the average download time. To achieve this goal, the seeders should remain active for some time after getting a copy of the file. Hence, it is important to calculate the *optimal seeding time* that minimizes the system energy consumption. The optimal seeding time for popular files is calculated in [10], on the basis of the results derived in [40]. In [40] the authors refer to the scenario depicted in Figure III and develop a fluid model of the system, based on the following assumptions. Let $x(t)$ and $y(t)$ denote the number of leechers and seeders, respectively, at a given time *t*. New peers (i.e., leechers) arrive with a rate λ. Once a leecher turns into a seeder, it leaves the system with a rate γ, i.e., it waits a seeding time $\tau = 1/\gamma$ until it shuts down. Since all peers are modeled equally, they are assumed to have a mean download rate of *c* and a mean upload rate of μ. In general, at a given point in time, uploading peers only have a part of the file and, hence, sometimes they may not be able to upload chunks to other peers. This is modeled through the *effectiveness* parameter η, i.e., a real value in the range [0,1] that represents the fraction of content already available at the peer. Since seeders have the whole file, their effectiveness parameter is equal to 1. As shown in Figure III, the total data flow is limited by either the total upload or download capacity. As a consequence, the actual dataflow is given by the minimum of these two flows. Specifically, under the above-mentioned assumptions, at a given time *t* the total download flow is $cx(t)$, while the total upload flow is $\mu[\eta \cdot x(t) + y(t)]$. It can be shown that the average download time T, which corresponds to the average time a leecher needs to turn into a seeder, is given by the following expression [40]:

$$T = \max\left\{\frac{1}{c}, \frac{1}{\eta}\left(\frac{1}{\mu} - \frac{1}{\gamma}\right)\right\} \quad (1)$$

Equation (1) is exploited by the authors of [10] to analyze the energy effectiveness of BitTorrent. They assume that each peer consumes one unit of power. Under this assumption, if λ leechers arrive each time unit, and each leecher needs, on average, T time units to turn into a seeder, then the total energy consumed by all leechers is λT. Likewise, if τ denotes the average time a seeder waits until leaving the torrent (by definition, $\tau = 1/\gamma$), then the total energy consumed by all seeders is $\lambda\tau$. Hence, the average total energy E consumed by the system is

$$E = \lambda \cdot \max\left\{\frac{1}{c}, \frac{1}{\eta}\left(\frac{1}{\mu} - \tau\right)\right\} + \lambda\tau \quad (2)$$

Figure IV shows the average total energy E/λ as a function of the average seeding time τ. For low τ values the average total energy consumed by the system tends to decrease as the average seeding time increases. This is because if there are more seeders uploading chunks, the download time experienced by peers – and the corresponding energy consumption – decreases. However, when the average seeding time becomes larger and larger, the additional energy consumed by seeders becomes predominant and it is not compensated anymore by the decrease in the average download time. Hence, the total energy consumption of the system increases. The optimal seeding time that minimizes the total energy consumption of the system can be derived from (2). It yields:

$$\hat{\tau} = \frac{1}{\mu} - \frac{\eta}{c} \quad (3)$$

Based on (3), in order to minimize the energy consumption, ideally each seeder should remain active for a time equal to $\hat{\tau}$ after getting a complete copy of the file. In practice,

the approach based on the optimal seeding time has a number of limitations that make it unpractical. First, the optimal seeding time depends on a number of parameters that are not under the user's control and, in addition, they also vary over time. Hence, it cannot be calculated by each peer. Besides, the user is typically *selfish* and *lazy*. He/she is mainly interested in getting a complete copy of the file (possibly with a minimum download time). He/she is not interested in minimizing the overall energy consumption of the system, especially if this is achieved at the cost of an increase in his/her personal energy consumption. In practice, the peer typically leaves the torrent after *some time* the file has been downloaded. The actual seeding time depends on user's convenience, not on energy considerations.

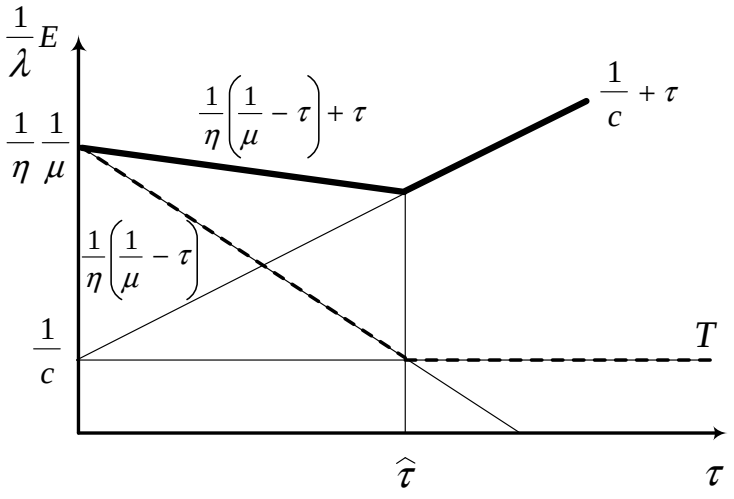

Figure IV. Average total power consumed by the system as a function of the average seeding time.

The optimal seeding time approach would be effective if it could be approximated so that the peer could calculate it at run time and automatically adjust the peer behavior in a transparent way for the user. This approach is exploited in *Green BitTorrent*, proposed in [18]. Green BitTorrent is an extended version of BitTorrent for energy efficiency and

takes an approach classified as traditional sleep/wakeup, according to our taxonomy in Section II. It allows peers to go to sleep when they are not downloading/uploading chunks, while remaining active members of the torrent. Green BitTorrent is backward compatible with the legacy protocol, i.e., Green BitTorrent peers can operate with traditional BitTorrent peers, even if the latter ones experience some performance degradation. Basically, Green BitTorrent extends the legacy protocol by introducing the concept of *sleeping* peer. In the legacy protocol peers establish TCP connections with their neighbors, which need to be always connected. In Green BitTorrent, a generic peer *P* in the torrent can appear to a tagged peer *T* in one of the following states [18]:

- *Connected*. Peer *P* has an active TCP connection with the tagged peer *T*. Hence, chunks can be uploaded and downloaded on the connection.
- *Sleeping*. Peer *P* has disconnected its TCP connection with the tagged peer *T*. The TCP connection must be re-established before chunks can be uploaded or downloaded.
- *Unknown*. The tagged peer *T* has received *P*'s address from the tracker, however, it does not know whether *P* is sleeping or awake.

To achieve its goals, Green BitTorrent relies on some additional mechanisms, namely an *inactivity timer*, a *connection timer*, and uses *wakeup messages*. The inactivity timer is reset whenever a download/upload activity occurs, and it is used to decide when the peer can go to sleep. The connection timer is used to discover that a peer has disconnected. Finally, a wakeup message is a special message used to wake up a sleeping peer [43]. With reference to a tagged peer *T* in the torrent, the following events can occur, which are managed as described below (see [18] for details).

- *Detection of a disconnected* peer. Upon discovering that a peer *P* has closed its TCP connection, *T* sets *P*'s state to sleeping.
- *Time-out of the inactivity timer*. As soon as the inactivity timer expires, *T* closes all its TCP connections and enters the sleep state.
- *Reception of a wakeup message*. Upon receiving a wakeup message from a peer *P*, the tagged peer *T* waits until its TCP connection with peer *P* has been re-established and, then, starts exchanging data with it.
- *Time-out of connection timer*. On a timeout of the connection timer, the tagged peer *T* checks the number of connected peers. If it is below a pre-defined threshold *max_connect*, it randomly picks up a peer *P* from the list received from the tracker, sends a *wakeup* message to *P*, and tries to open a TCP connection with *P*. If the last step was successful, then *P*'s state is set to connected. Otherwise, *P* is removed from the list.

It has been shown that Green BitTorrent can save about 25% of the energy, in comparison with legacy BitTorrent [18]. The drawback consists in an increased average download time experienced by peers which is, however, acceptable. Hence, Green BitTorrent is both practical and effective. However, it has a number of limitations. First, the wakeup message used by Green BitTorrent is typically implemented through a magic packet [43], which may not be available on some network technology (for instance, it may not be supported by WiFi network interface cards). In addition, there may be some privacy issues. For example, a remote peer is able to know whether your PC is active or sleeping. Also, a malicious user could wake up your PC, just to increase your energy

consumption. Finally, an attacker could suspend and wake up your PC with a high frequency to damage your hardware.

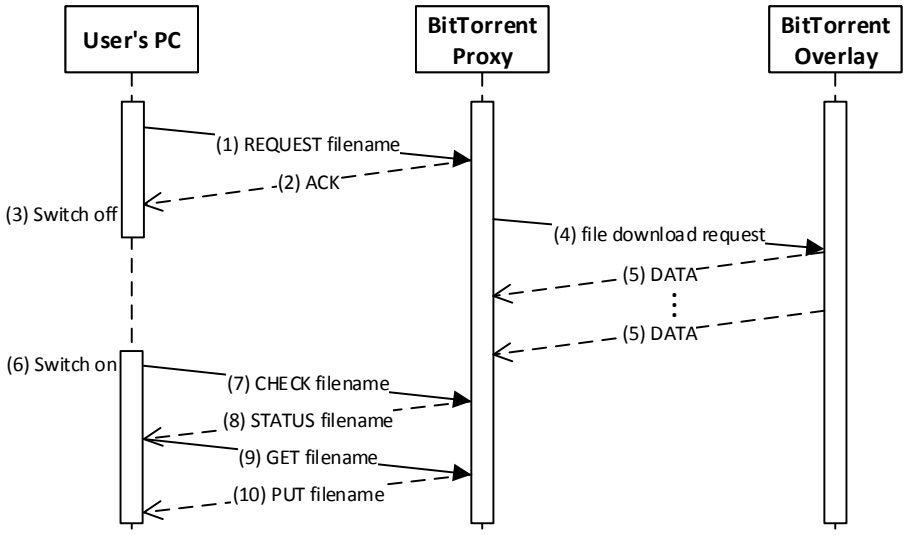

Figure V. Action performed in the EE-BitTorrent Protocol.

A completely different approach is used in *EE-BitTorrent* [8]. The proposed solution relies on a proxy-based architecture and, hence, belongs to the class of proxy-based sleep/wakeup systems, according to our taxonomy. It achieves energy efficiency by allowing a number of sleeping peers to be served by a *BitTorrent proxy*. Specifically, in EE-BitTorrent peers delegate the download task to the associated proxy, which performs the task on behalf of them. Most of the time, the requested file is already available on the proxy and can thus be immediately transferred to the user's PC. If this is not the case, the proxy participates in the BitTorrent overlay network as a regular peer, and takes care of the overall process. Therefore, the user's PC can be switched off during the download phase. The file will be transferred from the proxy to the user's PC later, when the user reconnects.

Figure V shows the actions performed by the various actors of EE-BitTorrent. Upon receiving a request from the user, the user's PC contacts the proxy and requests the desired file (step 1). The proxy acknowledges the request (step 2). If the requested file is already available in the proxy's local cache, it is immediately transferred to the user. Otherwise, the proxy starts downloading the requested file from the BitTorrent overlay network, acting as a regular peer and following the legacy BitTorrent protocol (steps 4-5). The user's PC can be switched off just after receiving the acknowledgement from the proxy (step 3). Later, when the user reconnects (step 6), he/she can check the status of the download process at the proxy (steps 7-8). If the file is completely available, it can be transferred from the proxy to the user's PC (steps 9-10).

Experimental measurements have shown that EE-BitTorrent performs very well in an institutional scenario where the BitTorrent proxy is located on the same high-speed network of users' PCs [8], while in residential scenarios the performance is strongly influenced by the uplink rate allowed by the access network [9]. When the uplink rate is low, the legacy BitTorrent protocol performs poorly, and EE-BitTorrent outperforms it in terms of average download time and energy consumption. The opposite occurs when the uplink rate is good. Motivated by these results, the authors of [9] have proposed an adaptive algorithm that dynamically selects the most efficient BitTorrent option (i.e., legacy or proxy-based), depending on the current operating conditions.

In terms of security, EE-BitTorrent is more robust than Green BitTorrent, provided that the BitTorrent proxy is a trusted machine. The basic idea of using a proxy for energy efficiency in BitTorrent has been highly influential in the design of later proposals, some of which are discussed below.

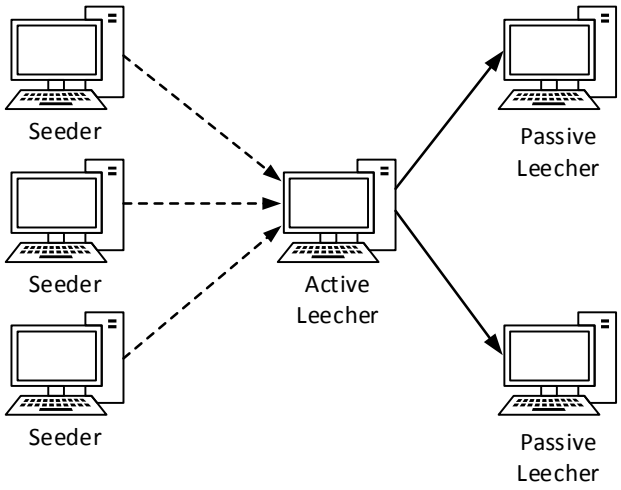

Figure VI. When the number of seeder is extremely low P2P file sharing may degenerate to a client-server file trasfer.

Download Sharing [10] also takes a proxy-based approach similar to that used in EE-BitTorrent, but it is mainly targeted to unpopular files. As well known, in BitTorrent a file is simultaneously downloaded by a multitude of peers. However, in case of unpopular files, the number of seeders is very low and it may happen that there are as many leechers as seeders. Under such conditions, the P2P paradigm degenerates to a client-server file transfer, where a single seeder serves one or more leechers. Hence, there is a dramatic performance degradation and a corresponding increase in the energy consumption. In such a case it would be more convenient to combine the upload capacity of seeders and avoid semantically redundant download flows. The basic idea behind Download Sharing is to randomly select a subset of leechers to act as *proxies*, and hibernate all the other leechers (see Figure VI). Proxies are called *active leechers* as they represent other *passive* leechers hibernating in the meantime to save energy. Active leechers should be selected based on their network performance related to seeders, i.e., the leecher with the fastest

link to seeders should be selected as proxy. Once downloads are finished, active leechers may send the content to yet passive leechers with the full upload bandwidth. The analysis in [10] shows that Download Sharing reduces the energy consumption of the overall system when the uplink bandwidth of the proxies is much larger than the download rate achieved from the BitTorrent overlay (which, for unpopular file, is typically very low).

Proxy-based solutions have been also used for content sharing in a mobile environment [11] [12] [13] [14] [15] [16]. The main goal here is to extend the lifetime of the mobile device. Different solutions have been proposed, which mainly differ in the location of the proxy. In [11] the authors propose *CloudTorrent* where the proxy resides on a cloud server. It is shown that, in comparison with a version of BitTorrent not using the proxy (*SymTorrent*), the proxy-based solution can significantly decrease the download time and reduce the energy consumption. In [12] the *router proxy* solution is presented where the proxy is hosted on the home router used by the user for residential Internet access. The home router is a good candidate for proxy due to its pervasiveness and stable energy consumption (i.e., independent from the operation load). The authors investigate the energy efficiency of the proposed approach by considering the memory limitation of the home router. It is shown that in both WLAN and 3G cellular networks the download speed of the torrent and the chunk size affect the energy consumption. In the router proxy solution, the effect of chunk size on energy consumption and the effect of upload/download ratio on proxy download rate are investigated [16]. In this study, two types of peers, namely limited (i.e. having limited repository) and regular peers (i.e. having sufficient repository) are defined. Then, the effect of increasing percentage of limited peers on the download time was analyzed.

In [13], a *distributed proxy* solution is proposed where multiple proxies are involved in the process of content download. In this solution, the aim is to increase the limited uplink rate of the home router, using multiple proxies, in order to meet the downlink capacity of mobile devices served. In the distributed proxy approach, the effect of increasing the number of proxies on energy consumption is investigated in [15]. It is shown that, after a certain threshold, adding more proxies to the system is not convenient since the aggregate proxy upload rate meets the download rate of the mobile devices. Energy consumption for various upload/download ratios in single and multiple proxy settings have been explored. The experiments have shown that the energy consumption of multiple proxies slowly decreases while the upload/download ratio is low (i.e., 0.2 - 0.4), and then becomes stable once the aggregate proxy upload rate meets the download rate of the mobile devices.

In [14], the router proxy presented in [12] has been extended to overcome the memory limitation of the home router. In the overlay network the router proxy behaves like an ordinary peer, but differently from the other peers, it is responsible for transferring the downloaded content to the mobile device. However, a problem arises due to limited memory of the router proxy. In the regular setting of the BitTorrent, when a peer downloads a chunk it declares that chunk as available for downloading. In the case of mobile device, the router proxy erases the content transferred to a mobile device in order to reuse the memory. That introduces the problem of deleted content that is no longer available for other peers. To overcome these issues, the authors divide the memory available on the proxy into two buffers, namely *upload* and *download buffers*. In the download buffer, downloaded chunks are kept but the content of the buffer is volatile. All

the uploaded chunks available to download are stored in the upload buffer. Each downloaded chunk transferred to mobile device stays in the buffer and is accessible by the swarm. This router proxy approach outperforms the torrent client in the mobile device from energy efficiency and download time point of view.

D. Energy Efficiency in Other File Sharing Protocols

After looking at the main solutions for increasing the energy efficiency in BitTorrent, in this section we consider other P2P file-sharing protocols. *Gnutella* is another very popular protocol and, after BitTorrent, the one that has received the largest attention from the research community, in terms of proposals for increasing its energy efficiency. It uses *query flooding* to find files in the overlay network. The standard protocol version (version 4.0) defines five message types: *Ping*, *Pong*, *Query*, *QueryHit*, and *Push* [44]. The Ping message is used to discover Gnutella hosts in the neighborhood and it is mainly used to build information about the neighborhood (reachable hosts, bytes shared, etc.). The Pong message is sent in response to a Ping. It includes information about the host sending it, such as uptime, amount of bytes shared and whether it is willing to receive connections or not. The Query message is used to find files. It is sent by the host to its neighborhood and, in turn, the neighbors forward the message to their neighbors, for up to a maximum number of times specified in the TTL field of the Query. The Query Hit message is the response from a Gnutella host that received the Query. It includes a file, or list of files, containing a combination of the keywords in the Query received. Finally, the Push message is used to download a file from a host that is behind a firewall. When a host receives a Push message, it opens a new TCP connection to the one that sent it. Through

this connection, the host can request the file from the firewalled host and consequently download it.

After version 4.0, the version 6.0 was released. The main modification of this version is the introduction of *ultrapeers* [5]. In this Gnutella version a peer can be either a *leaf* or an *ultrapeer*. A leaf connects to ultrapeers and shares with them the list of shared files. At first, and by default, all peers are leaf peers when they connect to the network. There are different ways to select peers to become ultrapeers in different Gnutella applications. The most common way is to use selection criteria such as the time the peer has been connected to the network, or the upload/download speed available to the peer. If the criterion is satisfied then the peer advertises itself as an ultrapeer to the network.

A proxy-based solution for energy efficiency in Gnutella has been proposed in [6]. The authors design a P2P power management proxy, operating in a low-power microcontroller. The proxy can be co-located within the host (e.g., on an Ethernet NIC) or in another device (e.g., a LAN switch). It supports a subset of a Gnutella host capabilities, including initiating and accepting connections to and from neighbors, receiving and forwarding Query messages, generating QueryHit messages. Since the proxy has limited capacity to store shared files, it wakes-up the corresponding host in order to carry out file requests. For adequately maintaining the query messaging between the proxy and host, some state information is essential. These are power state of the host (i.e. powered-on or sleeping), shared file names, and IP addresses of the neighbor nodes. File names can be shared between the host and the proxy in the form of a Bloom filter [45]. TCP connections are transferred between host and proxy by reestablishment. Generally, the proposed method allows computers to go asleep when they are idle and be

woken up by proxy when a request for file – in the form of a HTTP GET – is received. The experimental results show that the proposed proxy-based approach outperforms the legacy approach, in terms of query forwarding rate and, in addition, the proxy-based approach provides significant energy savings. The authors claimed that if 25% of all P2P hosts in the US were to adopt this method, a saving of over $38 million (in 2007 value) in energy costs could be achieved.

Another proxy-based solution for Gnutella has been presented in [7], where the authors propose a method called *power–proxying* that consists in selectively proxying a subset of protocol semantics via a separate hardware controller on the NIC (Network Interface Card) of each PC. When a PC is in standby state, the proxy handles all the Gnutella packets except the file upload requests. To deploy this approach in the real-word, a specialized NIC needs to be developed, which is a major limitation of this work.

Energy efficient solutions have been proposed also for wireless/mobile environments. Since running P2P file-sharing applications in a wireless/mobile Internet entails different constraints compared with those in the traditional wired Internet, the authors of [46] propose a message reduction approach to reduce the average energy consumption and delay of Gnutella on top of MANETs. They employ a gossip protocol and use its inherent probabilistic forwarding behavior to reduce the number of messages. Furthermore, the authors prove that the lower load allowed the P2P network to consume up to 32% less energy. For the same environment, Mawji et al. [47] propose a path selection algorithm for downloading files, using the Gnutella protocol, with constraints in energy costs and downloading time. The authors consider the problem of how client peers select not only which servers to download files from, but also which transitional nodes to choose that

constitute a path to reach the servers. To this end, they formulate a path selection algorithm that uses mixed integer linear programming to produce the optimal sets of paths for clients to select.

IV. Energy Efficiency in P2P Epidemic Protocols

Designing energy efficient epidemic (gossip-based) protocols and services has become significant due to their wide usage in large scale distributed systems. Several distributed services such as reliable multicasting [48][49][50], aggregate computation [51], frequent items discovery [52], overlay topology construction, failure detection, P2P streaming and data dissemination use epidemic approaches. Key advantages of epidemic protocols are their simplicity, scalability and high fault-tolerance properties. Moreover, all peers have equal responsibilities, and hence the system is inherently load balanced. Energy awareness of epidemic protocols attracted attention and has been addressed in [53][54][55][26].

An epidemic algorithm consists of periodical rounds. In each round, each peer contacts one or a few peers (i.e. neighbors) to exchange states. The algorithm finishes in multiple rounds, and data is disseminated to the network like an epidemic disease [48]. As depicted in Figure VII, in epidemic protocols there are three communication styles used to exchange states, namely *push-based*, *pull-based*, and *push–pull*. In the push-based model, each peer chooses random peer(s) to send its state. In the pull-based model, each peer chooses random peer(s) to receive their states, and in the push–pull model, each peer chooses random peers both to send and receive states.

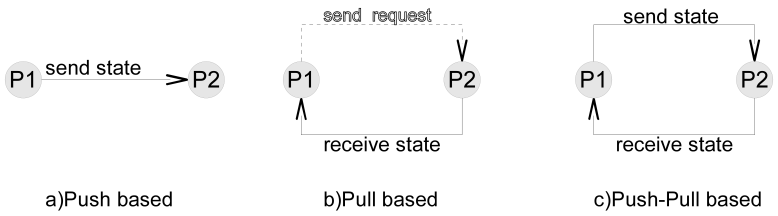

Figure VII. Communication styles in epidemic protocols

There exist two main classes of *epidemic algorithms*, namely *flat* and *hierarchical*. Flat algorithms include *basic* and *neighborhood* epidemics. The basic epidemic requires global knowledge of peer population and performs uniform gossiping, therefore it is not practical. On the other hand, neighborhood epidemic uses local knowledge which is more practical and performs gossiping with neighbors. Although neighborhood epidemic is better when compared to basic, it still has the problem of redundant communication. However, hierarchical epidemic makes use of structure among peers and aims to reduce communication overhead. In addition, it provides the possibility of active/passive peers to save energy.

In terms of their power usage, efficiency of the three above-mentioned models of epidemic protocols (i.e., basic, neighborhood and hierarchical epidemics) was first examined in [53]. Basic epidemics was found to be inefficient in its power usage. It has been shown that in neighborhood epidemics, peer's power consumption amount is independent of population size. On the other hand, for hierarchical epidemics, power usage increases with population size. However, this study evaluates different epidemics through simulations only and provides results on latency and power (proportional to the gossip rate). Moreover, effects of gossip parameters such as fan-out and maximum gossip message size were not investigated.

In terms of energy efficiency, the reliability of epidemic protocols has been discussed in mobile ad hoc networks [54] and a gossip-based protocol has been proposed for wireless sensor networks [55]. In [54], based on packet delivery ratio, nodes with high delivery ratio are classified as active, and therefore the energy consumption is affected significantly with less packet drops. In [55], low energy consumption and fault tolerance are achieved by early detection of the aggregation convergence independent of changes in the network topology.

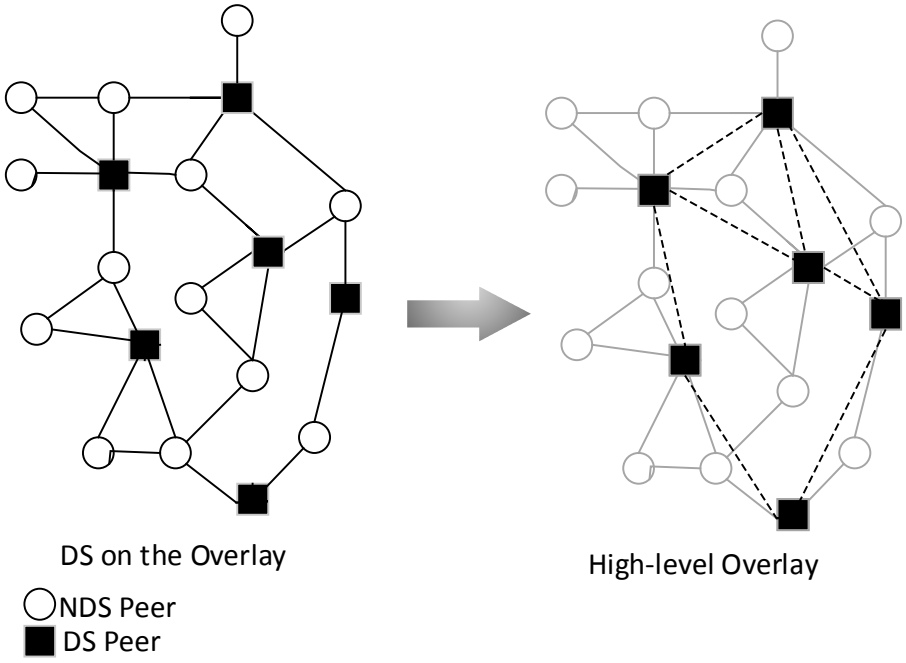

Figure VIII. Construction of high level overlay using dominating set

A recent study addresses power awareness features and develops energy cost model formulations for flat and hierarchical epidemics [26]. It also proposes energy cost models for generic peers using epidemic communication, and examines the effect of protocol parameters to characterize energy consumption. A novel *hierarchical epidemic* approach that uses dominating set while constructing the hierarchy is proposed. It utilizes the

benefits of both flat epidemic and hierarchical approaches. It uses only local knowledge and provides the possibility of active/passive peers to save energy. Robustness against peer failures is improved and the message overhead is significantly reduced thanks to the hierarchy. As a case study protocol, ProFID [52], an epidemic protocol for frequent items discovery in P2P systems, is used in [26]. Through large scale simulations on PeerSim, the effect of protocol parameters on energy consumption, comparison of flat and hierarchical epidemic approaches are performed.

In the hierarchical epidemic model [26], the dominating set idea is used to build a high-level overlay as illustrated in Figure VIII. Dominating Set (DS) can be defined as a subset B of a graph G=(V,E) such that every vertex in G is either in B or adjacent to a vertex in B. In this approach, a DS corresponds to a subset of peers such that a peer in the system is either in DS or a neighbor of a DS peer. There are two types of peers in the system, namely *dominating set (DS) peer* and *non-dominating set (NDS) peer*. The aim is to save energy by reducing the number of peers performing gossip operation. NDS peers are excluded from gossip operation. DS peers collect local states of NDS neighbors. Consequently, DS peers form a high level overlay topology on which gossiping is performed. There are two main advantages of this approach. First, NDS peers send their local state to one or multiple DS neighbors and, then, they switch to passive mode in which they just wait for the result to be announced by a DS neighbor. During this time period, they contribute reducing the energy consumption. Second, only DS peers participate in gossiping and, hence, the convergence time and message complexity of the epidemics reduces.

V. Conclusions

In this chapter, we have addressed the problem of energy efficiency in P2P systems and applications. Specifically, we have introduced a taxonomy that classifies the solutions proposed in the literature in three main classes, namely *Sleep/Wakeup* approaches, *Hierarchical* approaches and *Resource-allocation* based approaches. In addition, we have analyzed in detail how energy issues have been addressed in two relevant areas in P2P systems, namely file sharing protocols and epidemic protocols. With reference to file sharing protocols, we have emphasized the differences, in terms of energy consumption, between the P2P and client-server paradigms. In addition we have discussed the main solutions proposed for addressing the energy efficiency of two very popular protocols for P2P file sharing, i.e., BitTorrent and Gnutella. For epidemic protocols we have analyzed the differences, in terms of energy efficiency, between different classes of epidemic protocols.

While a number of solutions are already available, we believe that there is still room for research activities and improvements in both considered areas. We briefly discuss below some limitations of existing solutions. Most of the previous studies rely on simple energy models. For an accurate modeling, the energy cost should not only consider the energy consumption of individual peers (baseline energy consumption, energy consumed for processing, communication, etc.) but also the energy consumption related to network communication (e.g., energy consumed by routers, etc.). Obviously, more energy efficient protocols and applications could be implemented if the impact on the total energy consumption of the system was completely known.

Solutions proposed in the literature often exhibit limitations that reduce their applicability in a real environment. For instance, solutions that aim to determine the optimal behavior of peers in order to minimize the total energy consumption are not practical as, in reality, there is no way to force users towards an optimal behavior. However, also adaptive solutions are not exempt from limitations. For instance, solutions based on sleep/wakeup mechanisms –to turn on/off peers when they are not necessary – are prone to security and privacy problems, as emphasized in Section III. Finally, proxy-based solutions, although effective in certain scenarios, may not be very effective in other contexts. In the design of proxy-bases techniques a key point is the proxy location, which may significantly impact the energy efficiency of the overall system (as well as its robustness in terms of security/privacy). The use of low-power low-cost private proxies – instead of dedicated computers serving a large number of PCs – may be very appealing in certain contexts. We believe that the potentialities of such private proxies for energy efficiency in P2P have not been thoroughly investigated so far. However, a serious analysis would require considering economics costs in addition to energy efficiency.

References

[1] Y. Carlinet, H. Debar, Y. Gourhant, L. Mé, “Caching P2P Traffic What are the Benefits for an ISP?”, Proceedings Ninth International Conference on Networks (ICN), Menuires, France, April 11-16, 2010.

[2] H. Schulze, K. Mochalski, “IPOQUE – Internet Study 2008/2009”, Leipzig, Germany.

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

[4] K. Kant, “Challenges in distributed energy adaptive computing”, SIGMETRICS Performance Evaluation Review, vol. 37, pp. 3–7, January 2010.

[5] “Gnutella Protocol Specification Version 0.6.” [Online]. Available: http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[6] M. Jimeno, K. Christensen, “A Prototype Power Management Proxy for Gnutella Peer-to-Peer File Sharing”, Proceedings of IEEE Conference on Local Computer Networks (LCN), Dublin, Ireland, October 15-18, 2007.

[7] P. Purushothaman, M. Navada, R. Subramaniyan, C. Reardon, and A. George, “Power-Proxying on the NIC: A Case Study with the Gnutella File-Sharing Protocol,” in Proceedings 31st IEEE Conference on Local Computer Networks, 2006, no. 0519951, pp. 519–520.

[8] G. Anastasi, I. Giannetti, A. Passarella, “A BitTorrent Proxy for Green Internet File Sharing: Design and Experimental Evaluation”, Computer Communications, Vol. 33, N. 7, pp. 794-802, May 2010.

[9] I. Giannetti, G. Anastasi, and M. Conti, “Energy-efficient P2P file sharing for residential BitTorrent users,” in IEEE Symposium on Computers and Communications (ISCC), 2012, pp. 524–529.

[10] H. Hlavacs, R. Weidlich, T. Treutner, “Energy Efficient Peer-to-Peer File Sharing”, Journal of Supercomputing, available online at http://dx.doi.org/10.1007/s11227-011-0602-8.

[11] I. Kelenyi, J. Nurminen, "CloudTorrent – An Energy-Efficient BitTorrent Content Sharing for Mobile Devices via Cloud Service", Proceedings of IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA, January 9-12, 2010.

[12] I. Kelenyi, A. Ludanyi, J. Nurminen, I. Pusstinen, "Energy-efficient Mobile BitTorrent with Broadband Router Hosted Proxies", Proceedings IFIP Wireless and Mobile Networking Conference (WMNC), Budapest, Hungary, October 13-15, 2010.

[13] I. Kelenyi, A. Ludanyi, J. Nurminen, "BitTorrent on Mobile Phones – Energy Efficiency of a Distributed Proxy Solution", Proceedings International Green Computing Conference (IGCC), Chicago, USA, August 15-18, 2010.

[14] I. Kelenyi, A. Ludanyi, J. Nurminen, "Energy-efficient BitTorrent Downloads to Mobile Phones through Memory-limited Proxies", Consumer Communications and Networking Conference (CCNC), 2011.

[15] I. Kelenyi, A. Ludanyi, J. Nurminen, "Distributed BitTorrent Proxy for Energy Efficient Mobile Content Sharing", Wireless Personal Multimedia Communications (WPMC), pp.1-5, Oct 2011.

[16] I. Kelenyi, J. Nurminen, A. Ludanyi, T. Lukovszki, "Modeling Resource Constrained BitTorrent Proxies for Energy Efficient Mobile Content Sharing", Peer-to-Peer Networking and Applications, vol. 5, pp. 163–177, 2012.

[17] G. Lefebvre, M. J. Feeley, "Energy efficient peer-to-peer storage", in Technical Report (TR-2003-17), Department of Computer Science, University of British Columbia, 2003.

[18] J. Blackburn, K. Christensen, "A Simulation Study of a New Green BitTorrent", Proceedings of International Workshop on Green Communications (GreenComm), Dresden, Germany, June 2009.

[19] Wake on LAN Technology, White Paper: Rev 2 – June 1, 2006

[20] Y. J. Lee, J.-H. Jeong, H. Y. Kim, and C. H. Lee, "Energy-saving set top box enhancement in bittorrent networks," in Network Operations and Management Symposium (NOMS), IEEE, April 2010, pp. 809–812.

[21] M. Forshaw, N. Thomas, "A Novel Approach to Energy Efficient Content Distribution with Bittorrent", Computer Performance Engineering, Springer Berlin Heidelberg. pp. 188-196, 2013.

[22] G. Jourjon, T. Rakotoarivelo, and M. Ott, "Models for an energy efficient p2p delivery service," in Parallel, Distributed and Network-Based Processing (PDP), 18th Euromicro International Conference, Feb 2010.

[23] J.-S. Han, J.-W. Song, T.-H. Kim, and S.-B. Yang, "Double-layered mobile p2p systems using energy-efficient routing schemes", in Telecommunication Networks and Applications Conference (ATNAC), Dec 2008.

[24] T. Enokido, A. Aikebaier, and M. Takizawa, "A model for reducing power consumption in peer-to-peer systems", Systems Journal, IEEE, vol. 4, no. 2, pp. 221 –229, June 2010.

[25] A. Aikebaier, T. Enokido, M. Takizawa, and S. Deen, "Energy-efficient agreement protocols in p2p overlay networks", in Distributed Computing Systems Workshops (ICDCSW), IEEE 30th International Conference on, June 2010.

[26] O. Ozkasap, E. Cem, S. Cebeci, T. Koc, “Flat and Hierarchical Epidemics in P2P Systems: Energy Cost Models and Analysis”, accepted to Future Generation Computer Systems journal, Elsevier Science, 2013.

[27] T. Koc, E. Cem, and O. Ozkasap, “Dominating-set based and power-aware hierarchical epidemics in p2p systems”, in 2nd International Conference on Energy-Efficient Computing and Networking, 2011.

[28] A. Chandrasekar, K. Chandrasekar, H. Ramasatagopan, and A. R. Rafica, “SMC: An Energy Conserving P2P File Sharing Model for Mobile Devices”, Data Engineering for Wireless and Mobile Access (MobiDE), New York, pp.66-73, 2012.

[29] M. Raj, K. Kant, S. K. Das, “Energy Adaptive Mechanism for P2P File Sharing Protocols”, Euro-Par 2012: Parallel Processing Workshops, Lecture Notes in Computer Science, vol. 7640, pp. 89-99, 2013.

[30] P. Zhang, B. E. Helvik, “Towards Green P2P: Understanding the Energy Consumption in P2P under Content Pollution”, IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (GREENCOM-CPSCOM), 332-337, 2010.

[31] T. Enokido, A. Aikebaier, and M. Takizawa, “Process allocation algorithms for saving power consumption in peer-to-peer systems”, *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 6, pp. 2097 –2105, June 2011.

[32] A. E. Garcia, R. Weidlich, L. R. de Lope, K. D. Hackbarth, H. Hlavacs, and C. S. Leandro, “Approximation towards energy-efficient distributed environments,” in

Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools), 2010.

[33] H. Hlavacs, R. Weidlich, and T. Treutner, "Energy saving in future home environments," in IFIP Wireless Days, Nov 2008.

[34] H. Hlavacs, K. A. Hummel, R. Weidlich, A. M. Houyou, and H. De Meer, "Modelling energy efficiency in distributed home environments", Int. J. Commun. Netw. Distrib. Syst., 4, 2, 161-182, January 2010.

[35] K. Leibnitz, T. Hoßfeld, N. Wakamiya, and M. Murata, "Peer-to-Peer vs. Client / Server: Reliability and Efficiency of a Content Distribution Service", Managing Traffic Performance in Converged Networks, pp. 1161–1172, 2007.

[36] S. Nedevschi, S. Ratnasamy, and J. Padhye, "Hot Data Centers vs. Cool Peers", in HotPower'08 Proceedings of the conference on Power aware computing and systems, 2008.

[37] A. Sucevic, L. L. H. Andrew, and T. T. T. Nguyen, "Powering down for energy efficient peer-to-peer file distribution," ACM SIGMETRICS Performance Evaluation Review, vol. 39, no. 3, p. 72, Dec. 2011.

[38] L. Andrew, A. Sucevic, T. Nguyen, "Balancing Peer and Server Energy Consumption in Large Peer-to-Peer File Distribution Systems", Proceedings of IEEE Online Conference on Green Communications (GreenCom), Sept.26-29, 2011.

[39] J. Kurose, K. Ross, "Peer-to-Peer Applications", in Computer Networking. A Top-Down Approach, IV Edition, Addison Wesley, 2007.

[40] D. Qiu, R. Srikant, “Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks”, Proceedings of ACM Sigcomm 2004, Portland, Oregon, USA, pp 367–378.

[41] R. Kumar and K. Ross, “Peer-assisted File Distribution: The Minimal Distribution Time”, Proccedings of IEEE Workshop on Hot Topics in Web Systems and Technologies, 2006.

[42] G. Ezovski, A. Tang, L. Andrew, “Minimizing Average Finish Time in P2P Networks”, Proceedings of IEEE INFOCOM, Rio de Janeiro, Brazil, April 19-25, 2009.

[43] Magic Packet Technology, White Paper, Publication# 20213, Rev: A, Amendment/0, November 1995.

[44] “Gnutella Protocol Specification Version 0.4, 2001.” [Online]. Available: URL: http://www9.limewire.com/developer/ gnutella_protocol_0.4.pdf.

[45] M. Jimeno, K. Christensen, and A. Roginsky, “A Power Management Proxy with a New Best-of-N Bloom Filter Design to Reduce False Positives,” in IEEE International Performance, Computing, and Communications Conference, 2007, pp. 125–133.

[46] D. Neves da Hora, D. R. Macedo, J. M. S. Nogueira, and G. Pujolle, “Optimizing Peer-to-Peer content discovery over wireless mobile ad hoc networks”, in 9th IFIP International Conference on Mobile Wireless Communications Networks, 2007, pp. 6–10.

[47] A. Mawji and H. Hassanein, “Optimal path selection for file downloading in P2P overlay networks on MANETs,” The IEEE symposium on Computers and Communications, pp. 1133–1138, Jun. 2010.

[48] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y. Minsky,“Bimodal Multicast”, ACM Transactions on Computer Systems 17 (2) pp. 41–88, 1999.

[49] O. Ozkasap, Z. Genc, E. Atsan, “Epidemic-based Reliable and Adaptive Multicast for Mobile Ad hoc Networks”, Computer Networks 53 pp. 1409–1430, 2009.

[50] P. Eugster, R. Guerraoui, S.B.Handurukande, P. Kouznetsov, A. M. Kermarrec, “Lightweight Probabilistic Broadcast”, ACM Trans. Comput. Syst. 21 (4) pp. 341–374, 2003.

[51] M. Jelasity, A. Montresor, O. Babaoglu, “Gossip-based aggregation in large dynamic networks.” ACM Trans. Comput. Syst. 23(3), pp. 219-252, 2005.

[52] E. Cem, O. Ozkasap, “ProFID: Practical Frequent Items Discovery in Peer-to-Peer Networks”, Future Generation Computer Systems, 29 (6), pp.1544–1560, 2013.

[53] R. van Renesse, Power-Aware Epidemics, Proc. of IEEE Symposium on Reliable Distributed Systems, 2002.

[54] S. Rajeswari, D. Y. Venkataramani, “An Adaptive Energy Efficient and Reliable Gossip Routing Protocol For Mobile Adhoc Networks”, International Journal of Computer Theory and Engineering, 2 (5), pp.740–745, 2010.

[55] S. Fauji, K. Kalpakis, “A gossip-based energy efficient protocol for robust in-network aggregation in wireless sensor networks”, in Pervasive Computing and

Communications Workshops (PERCOM Workshops), IEEE International Conference on, 166 –171, 2011.