

Evaluating WUW, a service to enhance users' satisfaction in Content-Based Peer-to-Peer Networks

Raziel Carvajal-Gómez, Patricia Serrano-Alvarado

▶ **To cite this version:**

Raziel Carvajal-Gómez, Patricia Serrano-Alvarado. Evaluating WUW, a service to enhance users' satisfaction in Content-Based Peer-to-Peer Networks. Grid 5000 Winter School, Dec 2012, Nantes, France. hal-01362309

HAL Id: hal-01362309
https://hal.science/hal-01362309

Submitted on 8 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Evaluating WUW, a service to enhance users' satisfaction in Content-Based Peer-to-Peer Networks

Raziel Carvajal-Gómez and Patricia Serrano-Alvarado
{$Raziel.Carvajal-Gomez, Patricia.Serrano-Alvarado$}@univ-nantes.fr

Laboratoire d'Informatique de Nantes Atlantique (LINA) - Université de Nantes
2, rue de la Houssinière, 44322 Nantes, France

Abstract

Nowadays, Peer-to-Peer (P2P) architectures are becoming more popular in content delivery applications thanks to their valuable characteristics as scalability, performance and low maintenance costs. In those systems, peers share their resources automatically (bandwidth, storage, etc.) and not only download content but also upload content to other peers organized in a neighbourhood. Each peer' neighbourhood is based basically on QoS-related parameters (available bandwidth, number of connections, etc.) and the amount of exchanged content. We consider that peers are under control of users that are autonomous and free persons having rights, preferences and interests. As users' resources are the richness of P2P systems, we think it is important to satisfy their preferences beyond the QoS. In this paper we present first experimental results of WUW (What Users Want), a service located on top of a P2P layer and proposed to satisfy users' preferences during content exchange. In the current implementation we use the BitTorrent protocol for measuring to which extent users' preferences influence the P2P behaviour when WUW is used. We describe how the experimental scenarios are built using the resources provided by Grid'5000. Our preliminary results are encouraging because they show a low overhead of WUW on the global content sharing performance.

1 Context and motivation

Content Delivery Networks (CDN) [1] distribute content to end users as files (multimedia, software, documents), live-streaming, on demand streaming, etc. Peer-to-Peer (P2P) architectures are increasingly used in CDN because traditional client-server architectures generate high distribution and maintenance cost, whereas in P2P systems those costs are almost negligible. Besides, P2P architectures are more performing than client-server ones when high demanded content has to be distributed in short period of time.

P2P systems are highly scalable because in those systems peers share their resources automatically (bandwidth, storage, etc.) and not only download content but also upload content to other peers. Thus the more peers are in the system, the more resources the system has and the more performing it is. The P2P architecture is built on top of a physical network. Peers are organized in overlays (neighborhoods) composed of peers sharing the same resources, for instance, peers downloading the same file or watching the same TV program.

We consider that peers are under control of users that are autonomous and free persons having rights, preferences, interests, etc. As users' resources are the richness of P2P systems, we think it is important to satisfy their preferences concerning the usage of their resources. For instance, one user can prefer to share a movie with new users or just with users from France in a P2P network. By letting the users express their preferences and interests, we make it possible for them to express what they want in terms of a $strategy$. The concept of strategy defines the basic way for a user to determine which neighbors she considers interesting to trade with.

In general, CDN applications based on P2P architectures do not take into consideration user preferences, but only QoS-related parameters. In this paper we evaluate WUW (What Users Want) [2], a service that runs on top of unstructured P2P systems, which main goal is to allow the users to strategically impact their local neighborhoods, according to their personal preferences. The current implementation of WUW is located on the top of BitTorrent [3],[4], [5], an efficient P2P protocol for content distribution. We measure to which extent WUW considers users' preferences in a BitTorrent (called BT in the following) overlay and the overhead cost of WUW in BT. Experiments were conducted on the French testbed Grid'5000 [6] and our preliminary results show a low impact on the content sharing performance.

This paper is organized as follows. Section 1 resumes the main objectives of WUW, its architecture and the current implementation. Section 3 shows how experiments were conducted and our current results. Finally, Section 4 concludes and describes our ongoing work.

2 WUW (What Users Want) quick overview

The main objective of WUW is to allow users' preferences to influence the P2P content exchange process. With WUW, a user expresses her preferences and interests which are the inputs of her *strategy* to exchange content in the system. The

user strategy is used by WUW to build her neighborhood as close as possible to her preferences. Thus, instead of having an overlay based only on QoS constrains, as usual in P2P systems, through WUW, users have overlays based also on their personal preferences.

This section presents the key concepts of WUW (cf. Section 2.1), how its architecture works (cf. Section 2.2) and the implementation details over BT (cf. Section 2.3)

2.1 WUW overview

In WUW, the *strategy* is an abstract concept. Strategies can be application-based, content-based, by group of users, by user, etc. As in real life, strategies may evolve depending on the context. As a very simple example, a user may like movies of action and her strategy can be to exchange content in the P2P system with users that like same movies, but also with those living in Nantes. Through a strategy, a user can calculate how much others are interesting for her and thus define which are her intentions toward others. More formally, a

- $preference\ p \in P$ is represented by the couple $p = (label, value)$. For instance, movies preferences can be expressed by the couples $(category, drama)$ or $(category, horror)$.
- $strategy$ is a function $s : P \rightarrow \mathbb{R}$ that maps preferences into real numbers, these numbers are called $intentions$.

In a P2P context, a user, represented by a peer, acts simultaneously as a client and as a server. Intentions quantify, in a compact way, the attitude of a user over others, that is, to which extent a local peer prefers exchange with remote peers. User intentions as client and as server can be different, so WUW may need to calculate intentions twice, this actually depends on the strategy.

Once intentions are calculated, WUW performs a local ranking of peers by assigning scores to a locally known set of remote peers. To calculate meaningful scores, each user should know something about others, like for example their intentions. To spread this information, WUW needs an epidemic protocol. Thus, scores are calculated for instance by combining the local peer' intentions and the remote peers' intentions. The local peer is free to weight with a higher value its intentions or remote peers' intentions, that is, to which extent the local user takes into account her preferences versus neighbors' preferences. Then, WUW gives to the P2P application the n best scored peers, in this way the overlay is influenced by user' preferences.

Additionally, to evaluate to which degree the P2P application considers the local user preferences during the content exchange, WUW computes a $feedback$ periodically. To be able to compute the feedback, WUW needs information about the content exchange (download/upload activities) made in the P2P application. For that, we assume that any content C can be logically split in a set of $items\ i_1, ..., i_n$ and the content exchange is made by item. The feedback measures are defined in terms of $satisfaction$ and $adequation$. They are computed periodically, every n seconds, for the last download/upload activities (those that have not been taken into account in the last computation). Like intentions, each measure is computed twice:

- the satisfaction as a client of a local user measures to which extent neighbors she prefers the most were chosen by the P2P application to give her content;
- the satisfaction as a server of a local user measures to which extent neighbors she prefers the most were chosen by the P2P application to receive her content;
- the adequation as a client of a (local) user measures to which extent pieces requested (and downloaded) by her were available for downloading among neighbors she prefers the most;
- the adequation as a server of a (local) user A_{server} measures to which extent pieces she has were requested (and uploaded) for uploading by neighbors she prefers the most.

Feedback values can vary between 0 and 1. Intuitively, for satisfaction, 1 denotes the best choices were made and the peer is completely satisfied by the P2P system. For adequation, 1 denotes the local peer is giving the best scores to peers it is exchanging with. Feedback measures have been explained above briefly, the reader interested in more details can find a detailed explanation in [2].

2.2 Architecture

WUW functionalities are organized in different modules, Figure 1 shows the WUW architecture. First of all, through a User Front-End (e.g., a web page), users give their preferences to the WUW User Interface Handler and the content that the P2P application will trade with. Secondly, WUW periodically receives data about two separate concerns: events during the dissemination paradigm (throgh the Communication module) and events during the content exchange (through the P2P Handler). Those information is necesary to apply the user strategy. Finally, WUW communicates to the User Front-End of users how their neighbors are ranked and the status of the P2P application through the feedback measures.

In the next, each WUW module is described.

UI Handler. This module receives the user preferences as input and shows the feedback measures though a simple web-based interface.

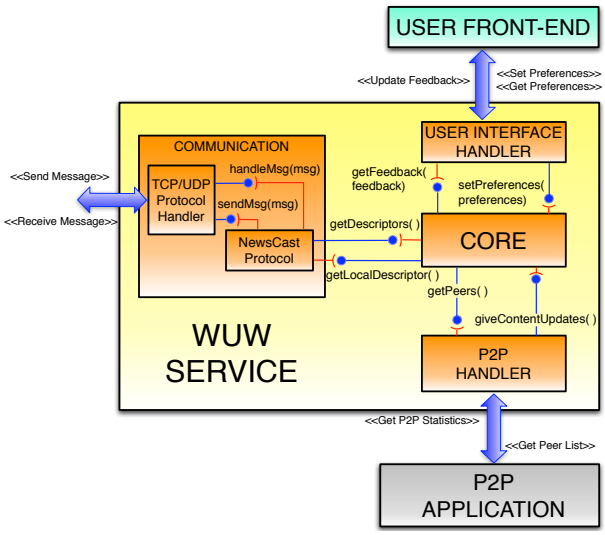


Fig. 1: WUW Architecture

Communication module. All the basic facilities to send and receive messages from remote WUW instances over TCP or UDP are implemented in this module, the epidemic protocol is implemented here as well. The information disseminated by the epidemic protocol is encapsulated in a *descriptor*. The content of such descriptor depends on the BT application used.

P2P Handler. This component is aware of the specific P2P application being used by the local peer. It performs two main tasks: **(i)** to get information about the upload/download events and **(ii)** to supply the P2P application with the set of ranked peers.

Core Module. The aim of this module is to coordinate WUW main functionalities. Periodically it:

1. gets the latest set of descriptors D from the Communication module;
2. gets the latest upload/download events E from the P2P Handler;
3. updates the local user intentions I towards each neighbor and the number of shared items in E;
4. updates the new ranking of users R with I and neighbors' intentions in D, then R is given to the P2P Handler;
5. updates the feedback measures F with D and I, then F is given to the UI Handler.

WUW is implemented as a multithread application thus any information exchange among the modules is asynchronous and concurrent.

2.3 Implementation

Currently, there is a prototype of WUW implemented in Java (JVM 1.7 or higher is required). This prototype runs on top of the BT Mainline 3.1.9 (Python implementation). The implementation issues to make interact WUW with the BT application are explained in the next.

BitTorrent as a P2P application. BitTorrent (called BT in this paper for short) is an efficient protocol for content distribution. A node is called peer and can be either seeder (this node has the complete content) or leecher (this node is still downloading the content). Thanks to a centralized node called tracker that provides the set of peers involved in the content distribution, each peer can communicate with each other. Files in BT are split into pieces and each piece is split into *chunks*, while chunks are the transmission unit of content, peers ask for pieces. When a peer wants to download a content it obtains from a web server a *.torrent* file. This file contains meta information about the content (name, size, number of pieces, etc.) and the tracker URL. The tracker is contacted and the pieces exchange starts among peers provided by it.

In order to perform the P2P Handler functionalities for a given content C, WUW needs two requirements: **(i)** to get metadata about the pieces exchange and **(ii)** to know which peers are interested in C. These requirements must be implemented in the P2P Handler independently of the P2P application.

Such that BT doesn't provide any facility for getting information about the pieces exchange, for the requirement **(i)** the Python Mainline 3.1.9 has been extended with an Status Collector. For the requirement **(ii)** the P2P Handler implements BT tracker basic functionalities, because this centralized node knows who is involved in the content distribution. Thus, according to how BT works, WUW is an intermediary between the tracker and the local peer. By launching a BT application, WUW has to acquire a .torrent file T and bootstraps as follows:

1. WUW gets content metadata and the tracker URL $btTrack_{URL}$ from T;

2. WUW replace $btTrack_{URL}$ by the P2P Handler URL in T;
3. the P2P Handler receives a request R of peers from the BT application;
4. WUW sent R to the BT tracker and the P2P Handler receives the first list L_{first} of peers;
5. WUW sends L_{first} to the BT application;
6. finally, the Core module executes its tasks periodically (cf. Section 2.2).

Classes for manipulating .torrent files and the BT tracker implementation were taken from the Java BitTorrent API [7].

Status Collector in BT. The aim of this component is to collect metadata about the upload/download events in the local BT application. Given that a BT peer sends/receives to/from its neighbors chunk requests, the Status Collector summarizes these requests by piece and by neighbor in a set of tuples. Each tuple has **(i)** the neighbor ID, **(ii)** the piece index, **(iii)** the type of request (two possible values: upload or download) and **(iv)** the state of request (three possible values: complete, ongoing or wrong). For instance, if the local peer P_1 is still downloading the piece number 5 and P_1 has received chunks from peers P_2 and P_3 previously, two tuples are created: $(P_2, 5, download, ongoing), (P_3, 5, download, ongoing)$. Per each content the BT local application trade with, a set of tuples $Tuples$ is created (based on the last pieces exchanges). Finally, the Status Collector answers a WUW request with $Tuples$.

Epidemic protocol. Apart from the facilities to send and receive messages over TCP and UDP, the Communication module has a basic implementation of the epidemic protocol Newscast [8]. This protocol disseminates periodically a local user $descriptor$ to a peer chosen randomly. A descriptor is a data structure formed by: **(i)** a unique user identifier, **(ii)** user intentions and **(iii)** the most recent downloads events. Descriptors are useful for calculating the score for ranking peers and the feedback (cf. Section 2.1). The accuracy of these calculations depends on the freshness of the received descriptors.

Strategy implementation. Currently, WUW implements a very simple test strategy based on two preferences. The first preference is *interest in peers* ($Interest$). Basically, each peer has a tag with the type of user it represents. There are three possible ordered values: $normal$, $touchy$ and $exigent$. The idea is to chose peers whose type is close in descending then in ascending order, i.e., a normal peer prefers peers like it then touchy peers then exigent peers; conversely, an exigent peer prefers peers like it, then touchy peers then normal peers. Given that intentions are numbers, to each value corresponds a number: $normal = 3$, $touchy = 2$ and $exigent = 1$.

Thus, let l and r be peers, where r is a remote peer (neighbor) of the local peer l. The $Interest$ vary in the range $[-1..1]$; if l and r have same interests the highest value is reached; conversely if l and r have opposite interests the lowest value is reached:

$$Interest = 1 - \frac{\mid Interest_r - Interest_l \mid}{2} \tag{1}$$

The second preference is the *number of downloaded and uploaded pieces*, accordingly Pcs_{client} and Pcs_{server}. The idea is to chose peers who provided/downloaded the most number of pieces without errors to/from the local peer. Pcs_{client} and Pcs_{server} are in the range $[1..-1]$. In the one hand, $Pcs_{client}(r)$ gives a proportion of the number of pieces l download from r ($DownloadEvents_r$) to the number of pieces l requested for downloading to r ($TotalDownloadEvents_r$). In the other hand, $Pcs_{server}(r)$ gives a proportion of the number of pieces l uploaded to r ($UploadEvents_r$) to the number of pieces r requested to l:

$$Pcs_{client}(r) = 2 * \frac{DownloadEvents_r}{TotalDownloadEvents_r} - 1 \ , \quad Pcs_{server}(r) = 2 * \frac{UploadEvents_r}{TotalUploadEvents_r} - 1 \tag{2}$$

Thus, intentions as a client $I_{client}(r)$ and as a server $I_{server}(r)$ of l towards r are given by next formulas where intentions are in the range $[1.. - 1]$.

$$I_{client}(n) = \frac{Interest + Pcs_{client}(n)}{2} \ , \quad I_{server}(n) = \frac{Interest + Pcs_{server}(n)}{2} \tag{3}$$

This strategy reaches a maximum value if two peers have the same interest and both peers have shared pieces without errors. For instance, if u has successfully downloaded every piece from n and both peers has the same interest, then $I_{client}(n) = 1$ which is the highest value.

3 Experimentation

An emulation of a real content delivery network was made using the resources of the French experimental testbed Grid'5000. The idea is to evaluate WUW against the impact in the performance of BT.

3.1 Test scenarios

We are interested in measuring the impact of WUW in a BT overlay. Two experiments were lunched in one set of nodes, in the first experiment each node has just the BT process and in the second experiment each node has two processes: WUW and BT (with the Status Collector). The set of nodes have the next configuration:

- 100 peers. One BT peer per node
- 10 seeders and 90 leechers
- Peer bandwidth: 256 Kb. It allows between 15 to 20 peer connections
- Content size: 112 MB (420 pieces)
- The tracker sends a list of 10 peers to BT peers

The parameters for WUW are the next:

- Calculation of intentions, ranking and feedback: every 8 seconds
- Dissemination time of the local user descriptor: 2 seconds
- Number of peers chosen by the dissemination protocol: 1
- Number of known descriptors by a local user: 50
- Number of the best ranked peers: 10

According to the current strategy implementation (cf. Section 2.3), in the second experiment, 50% of nodes are $normal$, 30% of them are $touchy$ and 20% of them are $exigent$.

3.2 Preliminary Results

Figure 2 shows that in average a leecher got the hole content in 384.21 seconds when using BitTorrent and in 364.02 seconds when using WUW with BitTorrent. Results show an improvement of 5.1% (20 seconds) for WUW+BT, this improvement is due to WUW has a light tracker implementation.

Figures 3, 4 and 5 present in average the ranking of the top 10 peers made by WUW versus the percentage of download. Each figure presents groups of peers of the same type, because in the current strategy (cf. Section 2.3) peers with similar interest are scored better. The results are obviously influenced by the experimental scenario. For normal peers (i.e., Figure 3) is easy to find peers like them because there are 50% of peers with a normal type. Touchy peers (i.e., Figure 4), after ranking all possible peers of same type, prefer exigent peers, but they are only 30% in the system so they are not always available. Exigent peers (i.e., Figure 5), after ranking peers of same type, prefer touchy peers.

We can notice that at the beginning of the download process, the ranking is not meaningful. That is because the ranking is based on the information disseminated by the epidemic protocol and, in average, after 10% of the download WUW has enough information (i.e., knows enough remote peers' intentions) to apply its strategy and rank peers.

3.3 Bash scripting in Grid'5000 is enough

Our experiments were conducted on the Grid'5000 testbed. Using bash-programming and some software utilities developed in the testbed, our experiments were performed as follows. Once the reservation of nodes is made using the `oarsub` command, and each node is executed in parallel through the `taktuk` command, the front-end node (in Grid'5000) waits till every node sends its local meta data (IP address, bash process ID, ports available for WUW and BT) via a `scp` connection. After that, the front-end sends a configuration file that contains a set of parameters for WUW and BT (bandwidth, seeder/leecher, URL of the BT tracker, content name, WUW parameters, etc.) via a `oarcp` request; when every leecher starts all the seeders have been in the overlay previously. Then on each node both processes collect statistics in log files, these files are useful for analyzing our results. Finally, when every leecher has downloaded the complete content log files are compressed and sent to the front-end.

Once the front-end has the results of every peer, a database is created for collecting statistics about: peers in the overlay, BT peers performance, pieces exchange in BT, dissemination of intentions, ranking and feedback measures. For analyzing and getting average values of our results, different queries to the database were performed.

Apart from the functions concerning WUW and BT, our current set of scripts are easy to adapt and can be used by other applications. We use 100 nodes in the Helios and Sol clusters from the Sophia site. Each node has 2 CPU's AMD at 2.2 GHz with 4 GB of memory.

4 Conclusions and Future Work

This work presented an evaluation to WUW, a service for content-based P2P networks that allows to enhance users' satisfaction in compliance with their preferences. We have presented first experimental results of the current WUW implementation, using BitTorrent as a P2P layer. According to our experiments and the current implementation of WUW, it

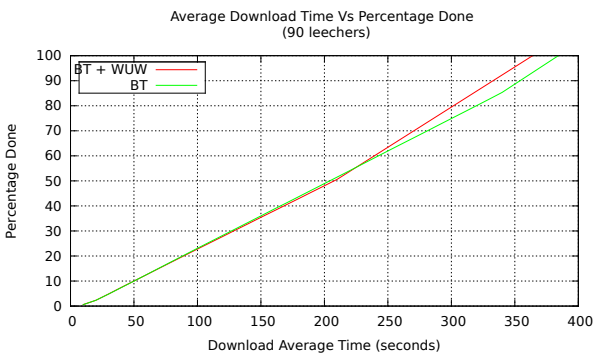


Fig. 2
Performance of WUW in terms of download average per second

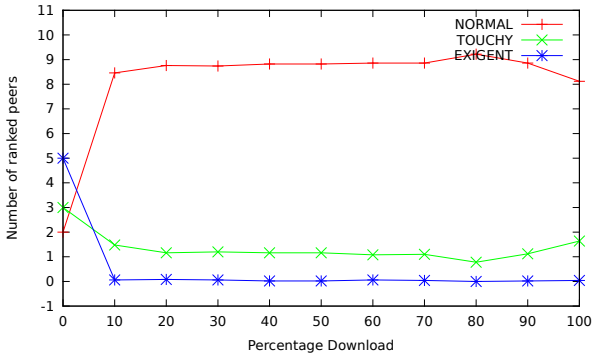


Fig. 3
Average ranking of *normal peers*

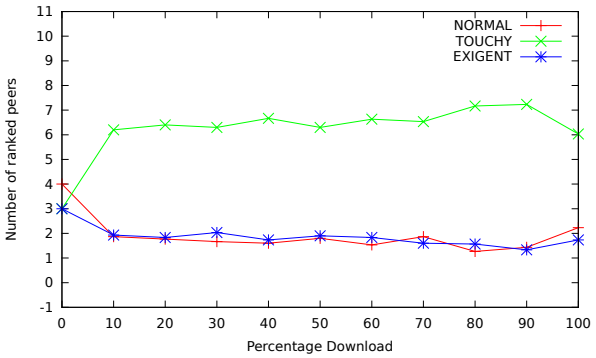


Fig. 4
Average ranking of *touchy peers*

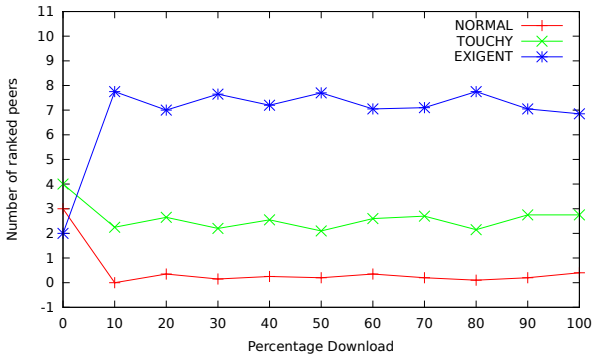


Fig. 5
Average ranking of *exigent peers*

can be clearly shown that WUW lets users to find others with similar preferences. Those experiments showed that without loosing performance WUW improves users' satisfaction. Additionally, the epidemic paradigm in our experimental scenario it is an efficient way for letting users know about what others users prefer.

Currently, we are trying to find a general strategy that can cope with different preferences at the same time. Find a concrete measure for evaluating to which degree users receive updated data through the dissemination paradigm, is one of our concerns too. Of course, we have to conduct experiments with a bigger number of peers and to consider scenarios with different configuration of preferences. The interpretation about our current statistics of the feedback is ongoing work. Furthermore, we have to measure if a BT overlay eventually evolves to another overlay where users' preferences are taken into account. The idea is to compare how many connection with peers, that WUW provides, the P2P layer make.

Acknowledgment. This collaborative work is realized in the context of the P2PWeb project supported by the Region Pays de la Loire. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

References

1. Buyya, R., Pathan, M., Vakali, A.: Content Delivery Networks. Lecture Notes in Electrical Engineering. Springer (2008)
2. Biazzini, M., Carvajal-Gomez, R., Perez-Espinosa, A., Serrano-Alvarado, P., Lamarre, P., Perez Cortes, E.: WUW (What Users Want): A Service to Enhance Users' Satisfaction in Content-Based Peer-to-Peer Networks. Technical Report hal-00744775, Université de Nantes (October 2012) 20 pages.
3. wiki.theor.orgy: Bittorrent protocol specification v1.0
4. Cohen, B.: The bittorrent protocol specification (February 2008)
5. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Workshop on Economics of Peer-to-Peer Systems, Berkley, CA, USA (June 2003)
6. Cappello, F., Caron, E., Dayde, M., Desprez, F., Jegou, Y., Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Quetier, B., Richard, O.: Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing. GRID '05, Washington, DC, USA, IEEE Computer Society (2005) 99–106
7. Dubuis, B.: Java Bittorrent API. `http://sourceforge.net/projects/bitext/` (February 2007)
8. Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.M., van Steen, M.: Gossip-based Peer Sampling. ACM Transactions on Computer Systems (TOCS) **25**(3) (August 2007)