BitTorrent Traffic Localization via Operator-related Information

Bartosz Polaczyk, Piotr Chołda
Department of Telecommunications, AGH University of Science and Technology, Kraków, Poland
{polaczyk|cholda}@kt.agh.edu.pl

***Abstract*—Applications based on peer-to-peer systems are oblivious to the underlying Internet topology. Chaotic connections between peers generate a large amount of traffic crossing carriers' borders, causing growth of operators' costs and have negative influence on performance (increased download time).**

In this paper, Yukka method is proposed to improve performance by localizing traffic on the basis of the operator and autonomous system related information. *YukkaPlugin*, an implementation of the Yukka method in one of the BitTorrent clients, uses for illustration RIPE NCC database responses mapping each IP address within Europe into an ISP network and country. The *YukkaPlugin* favors peers which are localized in an area close to a user. Applying the Yukka method results in shorter completion times achieved by an end-user.

***Index Terms*—Autonomous System (AS), BitTorrent, download time, inter-domain traffic, peer-to-peer (P2P) systems.**

I. Introduction

According to a current research study [1], peer-to-peer (P2P)-related load represents approximately 45-70% of the Internet traffic. What was first intended to be fun, that is a tool for file-sharing applications (Napster, Gnutella, BitTorrent), has now become an important branch of networking, related to generation of potenially large income. Such systems are known as overlay networks, forming its application-layer topology, frequently unaware of the underlying IP topology. P2P file-sharing systems are only one group of such networks. Others include multimedia streaming (SopCast), anonymous routing (Tor), or even Content Distribution Networks (CDNs, e.g., Akamai).

The unawareness of the underlying IP network related to P2P systems causes inoptimality of traffic flows in operators' networks [2], and is also responsible for undesirable inter-domain (inter-carrier) traffic generating large costs. Additionally, random connectivity observed in those networks is sub-optimal in the sense of the performance (download/completion time) that could be improved if P2P clients are connected having knowledge of their distance, network characteristics, geographical or networking localization. It was shown for the Gnutella network that only 2-5% of existing connections is formed within the same operator (ISP, Internet Service Provider), although 40% of users are located inside 10 largest ISPs [3]. There is much space for optimization.

Operators have given up using unfriendly methods of limiting P2P traffic and even start to cooperate with P2P application developers to improve their operation. The main current research trend related to this problem is to focus on the *localization/optimization of traffic* in P2P systems. It consists in an intelligent choice of application layer neighbors taking into account underlying IP conditions in the network layer. It matches topological proximity or operators policies (e.g., considering mutual operators' agreements) with the virtual links of P2P overlay networks. It is envisaged that at least two of involved parts interesting of this situation will be satisfied, that is: operators and P2P clients.

In this paper, we propose a method to localize traffic, where intervention from the operator's side is possible though not necessary, but modification of the P2P application is required. We take the user's perspective and focus mainly on its gain, i.e., the improvement of the download time. From this standpoint, localization of peers enables potential improvement of the transfer througput. This stems from the fact that connections between less distant nodes are less likely to be prone to congestions and delays. From this viewpoint it would be useful to choose peers from the same Local Area Network, then, from the same Autonomous System (AS), and, finally, the domain (group of ASes owned by the same operator) assuming that all those groups are in a relative topological proximity. Yukka, the proposed method, involves connections with the authority that gives a necessary information: it localizes the IP address by querying RIPE NCC (Réseaux IP Européens Network Coordination Centre, http://www.ripe.net) database. On the basis of the responses, Yukka assigns an IP address to the related carrier and country.

We focus on BitTorrent, a protocol for file sharing, used for finding peers that store content searched by an interested user. It uses a central server, *tracker*, that is unique for a selected shared content (identified by a hash number). It is found owing to a short `.torrent` file (usually available by WWW), identified with the shared content to simplify terminology. Each `.torrent` file defines an independent overlay network, known as a swarm. A tracker is responsible for informing a peer what are IP addresses (and opened transport layer ports) of other peers having the content associated with a selected `.torrent`. For one query, some random group of such peers is sent to the user. Then, the peers connect and share information on fragments of the content they have. As BitTorrent clients cannot have active uploading connections to all peers in a swarm, they actively use only a small number of such connections, where most of them is decided on the basis of the tit-for-tat strategy where data is sent to those peers, which offer the best download rate to the local peer. It is also possible

to find more attractive peers to optimize the download, but this is decided randomly (optimistic unchoking). Localization methods related to BitTorrent are focused on modification of this random behavior to suggest potentially attractive peers.

The next section presents the bibliography background for the problem. Section III introduces the core mechanism of the proximity seeking in the Yukka method. Next, Section IV gives details on how it is implemented in the networking environment. Section V focuses on the performed experiments that show that the proposed solution, yet very simple, can be useful. The last section summarizes our approach and details our concepts on further research.

II. Previous Work

A survey on works related to the choice of optimal peers from the viewpoint of the P2P nodes is presented in [4]. The earliest reasearch on locality was related to a specific overlay topology formation and operation due to the proximity-related data, especialy for structured overlays [5] like for instance popularly studied Pastry [6]. A solution based on the longest IP prefix matching was also proposed [7]. A similar but broader idea, called a biased neighbor selection [8] with a modified trackers, consists in choosing as many peers connected to the same ISP as possible. On the other hand, [9] proposes to gain the information that can improve the overlay formation on the basis of a DNS (Domain Name Service) lookup correlated with global CDN information. As clients connect CDN nodes on the basis of topologically and geographically optimized distance, it is likely that the peer nodes that would choose the same CDN node are also closely located. This method neither needs to modify trackers nor requires cooperation with an operator. It is realized using a commonly available service (DNS) needing instalation of a plug-in to Vuze, a BitTorrent client application.

The second large set of approaches is based on a strict cooperation between ISPs and overlay networks [10]. Two groups of solutions can be found here, that is locality awareness and network caching [2]. While the latter method can be performed by an operator alone, the former is done by peers with the carrier cooperation. Aggarwal et al. [11][1] propose introduction of 'oracle,' an ISP-owned facility that will provide overlay client with information that helps to localize P2P-based traffic. A similar approach, yet more grounded in the BitTorrent concept is known as P4P [12], where the usage of a specialized tracker provided for each AS is proposed. A working group *Application-Layer Traffic Optimization* (ALTO), gathering carriers and software companies, has been created within IETF to consume the efforts performed so far and to prepare a protocol that can be used between peer nodes and operator facilities [13]. For those methods we can obtain a better peer selection accuracy (because operator has usually a wider knownledge of network topology). However, they require additional ISPs' effort.

[1]They mention a possibility of using a method for localizing peers similar to our idea, but it is not evolved as the option is apparently treated as a temporary solution before ISPs can offer specialized services determining to which AS a selected peer is connected.

The method presented below is a trade-off of those two approaches. Localization is done on the peer's side, without involving ISP and using a more accurate database provided by RIR (Regional Internet Registry). For better results, operator can arrange a simple server, which speeds up the peer introducing process.

III. Yukka: General Idea

As most of trackers generate their responses as a randomly chosen subset of peers cooperating in a swarm, each peer can obtain a list of IP addresses (with ports) spread around the world. Unlike other oracle-like proposals, our method does not require to modify existing BitTorrent infrastructure. Instead, a new optional facility *YukkaTracker* is introduced. It applies a biased neighbor selection technique. However, the Yukka method can also work without this facility.

A peer exploiting the method, called here a *YukkaUser*, performs localization of each neighbor IP address (sent by a tracker) to determine if a particular peer exists in a close area. The localization employs `whois` protocol offered by RIR related to a particular region of the world, e.g., RIPE NCC in Europe or ARIN in North America. Based on information provided by the RIR database (country, netname, description and maintained_by fields) the method estimates how far a peer with a given IP address is. This is performed by computing the $similarity$ value. Algorithm 1 presents how the similarity value is calculated for users A and B. R^U_f denotes the value of field f of the RIR response corresponding to the IP address of user U.

Algorithm 1: Yukka calculation of $similarity$

$similarity \longleftarrow 0$;
if $R^A_{description} = R^B_{description}$ *or* $R^A_{netname} = R^B_{netname}$ **then**
  $similarity \longleftarrow 1$;
else if $R^A_{maintainted\_by} = R^B_{maintainted\_by}$ **then**
  $similarity \longleftarrow 0.75$;
else if $R^A_{country} = R^B_{country}$ **then**
  $similarity \longleftarrow 0.5$;

Each *YukkaUser* assigns other peers into one of the following three groups:

1) *littleYukka*, if $similarity$ is smaller than 0.5,
2) *middleYukka*, if $similarity$ is in the range $[0.5, 0.75]$,
3) *bigYukka*, if $similarity$ is greater than 0.75.

*littleYukka*s are long-distance peers, therefore, they are not recommended candidates to exchange data. To ensure that the Yukka method does not adversely affect the download speed, we let P2P application to contact with them on the native manner. On the other hand, connections with *middleYukka*s and *bigYukka*s are desired. Hence, Yukka strives to encourage a user to choose such peers.

Having a small study on geographical localization of members obtained from a typical tracker response, we realized that the average probability that a given peer in a typical swarm comes from the same country (Poland, here) is quite

small (less than 0.1%). The most popular shared files are related to swarms consisting of thousands of users spread around the world. Then, to increase probability of finding local users, we propose an optional facility known as *YukkaTracker*. It performs recurrent tracker queries to browse all peer IP addresses in a swarm on behalf of its clients, enabling to pick the most interesting candidates.

YukkaTracker keeps all peer addresses whose $similarity$ is not less than 0.5. Note that this $similarity$ is calculated from the *YukkaTracker* viewpoint, therefore to be useful for a *YukkaUser*, it must be located as close as possible to the *YukkaTracker*. According to our approach, this facility should be hosted by ISPs to let its client to receive closest peer addresses. IP addresses of *YukkaTracker*s can be obtained for instance, with DNS lookups related to hostname tracker.Yukkatracker.com.

IV. Yukka: Practical Implementation

A practical implementation of Yukka is developed as a Java-written plug-in to Vuze application to extend its native operation. *YukkaPlugin*, a plug-in dedicated for *YukkaUsers*, consists of 3000 lines of code. *YukkaTracker*'s plug-in, called a *YukkaTrackerPlugin*, contains 2000 lines of code. Current version of the both is available at http://bartosz.polaczyk.com/Yukka. Now, those plug-ins are aimed for European users as they work for IP addresses administrated by RIPE NCC. *YukkaPlugin* and *YukkaTrackerPlugin* query the RIR database with IP addresses obtained from trackers related to the downloaded contents. Afterwards, once they get new peer addresses, they consult RIPE NCC again. One may expect that working with small swarms may cause redundancy of queries. For this purpose, we introduced a cache storing last 1000 RIR responses.

Once *YukkaPlugin* finds a user with a high $similarity$, it attempts to connect to it as frequently as possible. Its IP address is marked as desired and enabled in a *favoritism* process. It consists in two actions. 1. In the case of a failed handshake process, the process tries to connect twice with such a peer in next 5 minutes. 2. It attempts to replace inefficient connections (apparently long-distance ones with poor download speeds) by connections with desired peers.

Unlike in the case of the native functionality, where each peer has to announce a Vuze-based tracker to be useful, our *YukkaTracker* can query other trackers to search on-line peers in the entire swarm. Every 10 seconds it queries the tracker and requests for maximum 50 peer addresses. To determine if a particular peer is placed in proximity, a localization procedure is the same as in the case of a *YukkaPlugin*, i.e., Algorithm 1 is applied. Due to the fact that no connection with peers is involved, *YukkaTracker* assumes that peers whose addresses are not sent by the original tracker during last 60 minutes have gone off-line (left the swarm). Querying the original tracker and finding peers is known here as *Peer Finding*. *YukkaUser* has two methods to obtain addresses stored at the *YukkaTracker*. Firstly, it can open a TCP connection (port 8345) or simply contact the tracker in a traditional way. Such two options for interfaces facilitate BitTorrent clients which do not implement Vuze's *YukkaPlugin* localization method, to also benefit from the *YukkaTracker*'s feature. The TCP communication scheme between a *YukkaUser* and a *YukkaTracker* is presented in Fig. 1. In the beginning, *YukkaUser* initiates a TCP connection and sends Yukka Request message **(1)**, which contains only a hash value of a given file. In case when the *YukkaTracker* has already registered this file (and is performing the *Peer Finding* process for it), it replies with a list of local peer addresses via Yukka Response message **(2)**. Otherwise, the *YukkaTracker* asks for a `.torrent` file **(3)** and after receives it **(4)**, automatically adds given `.torrent` into the tracker list. The amount of data that has to be transferred to determine a peer's proximity on the *YukkaUser*'s side is negligible. By default, the Vuze application asks a tracker for maximum 50 addresses and the interval between two announcements is usually longer than 20 minutes. Generally, the `whois` response is about 1 KB, so each downloaded file generates up to 150 KB of signaling traffic per hour. To investigate if *YukkaTracker* requires much more bandwidth, we examined a swarm with a large amount of users (more than 5000) logging the signalling traffic with the RIR database. It appeared that such a swarm generates 170 KB of upstream and 10 MB of downstream overhead per day.

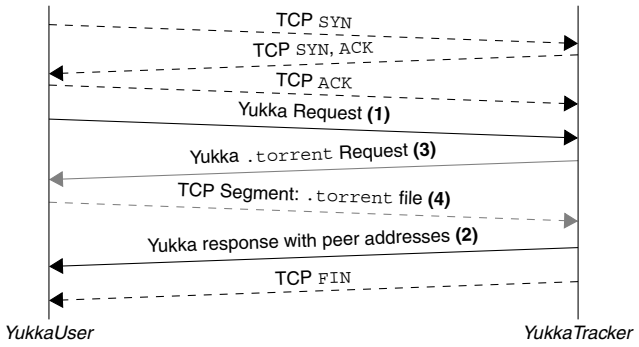


Fig. 1. *YukkaUser*-*YukkaTracker* communication scheme.

V. Experimental Study

The benefit of the Yukka method can be twofold: it reduces the average download time experienced by an end-user and may decrease the inter-ISP traffic, too. In our empirical tests we focused on the first feature, as we are able to study experimentally the client's perspective only. We believe that enabling benefits for customers is the only way to make the proposed approach popular in the real life. Each alleged improvement of the P2P protocol, which adversely impacts the user's performance relieving on the ISP's revenue, could be useless due to the fact, that no client is interested in applying it. In this section, we present results demonstrating the effectiveness of the Yukka method in a real BitTorrent network.

A. Experiment Environment

To determine the Yukka's impact on the average completion time, we examined different data downloads in two scenarios:

TABLE I
DATA ON TESTED DOWNLOADED FILES

File type	File size	Average swarm size (number of peers)
MP3 file (one song)	6 MB	2005
Entire music album	186 MB	1560
Disc image of an operating system (OS)	700 MB	75

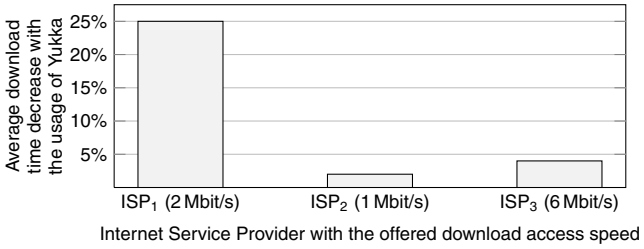


Fig. 2. Mean improvement of Yukka method for three different Internet service providers.

1. with a native Vuze operation, and 2. enabling *YukkaPlugin*. To compare the native- and Yukka-based completion times we introduce the improvement parameter I_Y defined as follows:

$$I_Y = \frac{t_N - t_Y}{\frac{t_N + t_Y}{2}} = 2\frac{t_N - t_Y}{t_N + t_Y}$$

where t_N is a native completion time and t_Y is a Yukka completion time.

A real overlay network is difficult to represent in a laboratory: it contains users behind Network Address Translation, firewalls and spread throughout the whole world with congested networks, etc. Therefore, we decided to perform experiments on existing swarms with real peers. To study the impact of different ISPs, we launched tests using Internet access from selected three operators offering services in Kraków. In our experiment we used three types of downloaded files, shown in Table I. The entire volume of the transferred data in tests equals 33 GB. It was obtained from 360 separate downloads. Below, only a fragment of results is presented. The remainder can be found at http://bartosz.polaczyk.com/Yukka.

B. Impact of an Internet Service Provider

The plug-in was tested while the client used separately access connections to three different Kraków's ISPs. We do not reveal their names, but call them ISP_1 (an international operator), ISP_2 (operating in the Kraków area only), and ISP_3 (a country operator). In Fig. 2 we present the improvement of the Yukka method depending on three ISPs' connections.

Firstly, we can observe that there is a considerable difference of the Yukka effectiveness between ISP_1 and two other ISPs. To find the cause, we tested the connections with the BitTorrent blocking detector [14]. Positive results of the tests confirm that ISP_1 throttles BitTorrent traffic (for more than 90%) when it crosses its network or country (we cannot distinguish it) borders. In this case, the Yukka method has a great potential to present its positive performance impact.

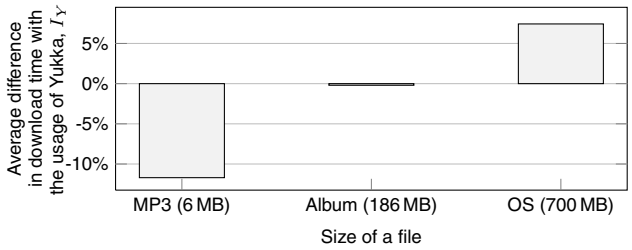


Fig. 3. Mean improvement of Yukka method for three types of file size.

Thus, using the ISP_1 Internet access connection, the Yukka method results in 25% completion time reduction. Contrary to ISP_1, ISP_2 and ISP_3 do not tamper with BitTorrent traffic, therefore, the average improvement is not so high, however, still oscillates around 2-4%. Furthermore, we noticed that with the slowest connections (ISP_2, 1 Mbit/s) a Vuze client often achieves the maximum transfer rate independently of the Yukka usage, thus a potential gain of the proposed method is limited by the access bandwidth.

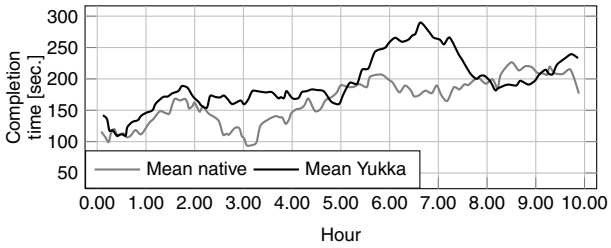
C. Impact of a Downloaded File Size

To estimate an impact of the downloaded file size, we chose to use Internet access offered by ISP_2 (the least benefical case) and examined three different files with various data sizes presented in Table I. The difference of the dowload time (I_Y) for the Yukka method is presented in Fig. 3. The negative value means that in fact we did not observe the improvement, but the deterioration.

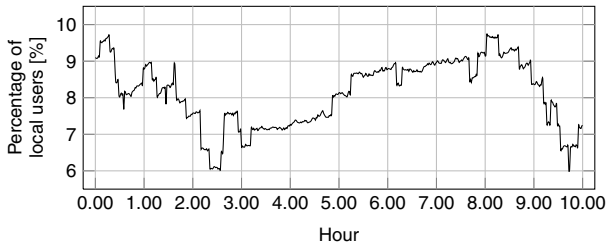
It is obvious, that in the case of short downloading times (related to small downloaded files), the Yukka method achieves worse results. The *YukkaPlugin* announces *YukkaTracker* for the first time after 60 seconds, thus the remaining time, when Yukka takes benefit of localization is negligible in comparison to entire completion time. Additonally, determination of peers proximity utilizes some bandwidth. Moreover, YukkaPlugin determines peers proximity immediately after tracker's announcement—only after completion of this process, it passes tracker's response to Vuze. At last, each peer has to be located before being introduced in Vuze, what in cases of long RIR database responses can give a negative effect. Those drawbacks are visible and express their malice only while downloading small files. For longer downloading times, the Yukka method seems to be dominating, thus we observe a positive improvement with larger data sizes.

D. Number of Local Users

To demonstrate how the number of local users affects the Yukka's effectiveness, we observed the download time difference over the 10 hour long test for downloading the MP3 file (again, the least benefical case, yet the largest swarm). Fig. 4(a) presents a graph of the temporary mean for the completion time. Fig. 4(b) shows the percentage of local users over a particular time period in the entire swarm. By analyzing this example, we can see that there is a strong



(a) Temporary mean values of download times.



(b) Percentage of local users in the entire swarm (*bigYukka*s and *middleYukka*s).

Fig. 4. Study of the improvement of the Yukka method downloading time for a MP3 file. Temporary mean values are calculated on the basis of measurements from the preceding several points.

correlation between the number of local peers and the Yukka's effectiveness on the end-user side. As we can observe, the single period, when the Yukka method reports a perceivable gain in comparison to the native client operation is 8.00-9.00AM. At this time the number of local users achieves its maximum. On the other hand, the decrease of closely located users immediately adversely affects the Yukka's performance.

VI. Summary and Further Research

In this paper, we proposed and evaluated benefits of employing the Yukka localization method in the BitTorrent environment. To determine if a given IP address is interesting, the RIR database response is involved. Choice of local users is beneficial for ISPs and end-users; as well as, implicitly, the overlay providers, if present. We performed an investigation of a real implementation for a Vuze client, a *YukkaPlugin* and *YukkaTrackerPlugin*, in the real Internet environment, using an Internet access connection from three existing ISPs. It shows that the proposed method results in a shorter completion time for end users. We can notice that the final benefit strongly depends on the amount of proximate peers, data size and ISP's policies on the P2P traffic. Overall, the Yukka method achieved 4.31% of the average improvement of completion time in our studies. Since most of existing approaches are investigated with simulations or using laboratory networks, it is difficult to compare obtained results with those achieved by other researchers. However, in [9] it is reported that the average download speed increases by 31% (in comparison to other peers) but the total end user download time is not estimated. On the other hand, results for P4P reported in [12] using real (PlanetLab) environment show about 10-20% of the download time improvement.

Further work can follow many avenues. The improvement of the software implementation by involving RIR databases other than RIPE NCC to enable this method to work worldwide, is necessary. Moreover, a distributed service of the *YukkaTracker*, where responsibilities of querying an original tracker can be passed to a selected user, picked by a specialized algorithm, can be developed. On the other hand, estimates how the Yukka method affects the inter-ISP traffic reduction should be performed. One may say that the results presented in the numerical section are achieved due to a larger number of all introduced peers for the Vuze client (*YukkaUser* utilizes addresses provided by original tracker and, in addition, from *YukkaTracker*). To investigate this issue, we envisage comparison of our results to the ones obtained on the basis of the scenario when the *YukkaTracker* keeps the same number of peers but the peer picking involves random selection.

Acknowledgements

The authors would like to thank Professor Andrzej Jajszczyk for his help while working on this paper.

References

[1] H. Schulze and K. Mochalski, "Internet Study 2008/2009," 2009, ipoque GmbH Whitepaper.

[2] A. Damola, V. Souza, P. Karlsson, and H. Green, "Peer-to-Peer Traffic in Operator Networks," in *Proc. IEEE 8th International Conference on Peer-to-Peer Computing P2P 2008*, Aachen, Germany, Sep. 8-11, 2008.

[3] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella Network," *IEEE Internet Comput.*, pp. 50–57, Jan./Feb. 2002.

[4] I. Rimac, V. Hilt, M. Tomsu, V. K. Gurbani, and E. Marocco, "A Survey on Research on the Application-Layer Traffic Optimization (ALTO) Problem," Dec. 2009, IETF Informational Draft (work in progress).

[5] K. P. Gummadi, R. Gummadi, S. D. Gribble, S. P. Ratnasamy, S. Shenker, and I. Stoica, "The Impact of DHT Routing Geometry on Resilience and Proximity," in *Proc. 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications SIGCOMM'03*, Karlsruhe, Germany, Aug. 25-29, 2003.

[6] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Topology-aware Routing in Structured Peer-to-Peer Overlay Networks," 2002, Microsoft Research Technical Report MSR-TR-2002-82.

[7] H. Le, D. Hoang, and A. Simmons, "A Self-Organizing Model for Topology-Aware Overlay Formation," in *Proc. IEEE International Conference on Communications ICC 2005*, Seoul, Korea, May 16-20, 2005.

[8] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang, "Improving Traffic Locality in BitTorrent via Biased Neighbor Selection," in *Proc. 26th IEEE International Conference on Distributed Computing Systems ICDCS'06*, Lisbon, Portugal, Jul. 4-7, 2006.

[9] D. Choffnes and F. Bustamante, "Taming the Torrent," in *Proc. ACM SIGCOMM Conference on Data Communication SIGCOMM 2008*, Seattle, WA, Aug. 17-22, 2008.

[10] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet Service Providers Fear Peer-Assisted Content Distribution?" in *Proc. 2005 Internet Measurement Conference IMC 2005*, Berkeley, CA, Oct. 19-21, 2005.

[11] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P Systems Co-operate for Improved Performance?" *ACM SIGCOMM Comput. Comm. Rev.*, vol. 37, no. 3, pp. 29–40, Jul. 2007.

[12] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in *Proc. ACM SIGCOMM Conference on Data Communication SIGCOMM 2008*, Seattle, WA, Aug. 17-22, 2008.

[13] J. Seedorf and E. W. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement," Oct. 2009, IETF RFC 5693.

[14] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi, "Detecting BitTorrent Blocking," in *Proc. 2008 Internet Measurement Conference IMC 2008*, Vouliagmeni, Greece, Oct. 20-22, 2008.