Computer Networks 56 (2012) 642–660

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Adaptive Search Radius – Using hop count to reduce P2P traffic ☆

Ricardo Lopes Pereira [a,*], Teresa Vazão [a], Rodrigo Rodrigues [b]

[a] INESC-ID/Instituto Superior Técnico, Technical University of Lisbon, Av. Professor Cavaco Silva, 2744-016 Porto Salvo, Portugal
[b] Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern and Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 14 May 2010
Received in revised form 8 April 2011
Accepted 24 October 2011
Available online 31 October 2011

Keywords:
Peer-to-Peer
File sharing
Traffic reduction
Performance improvement
Locality aware

ABSTRACT

Peer-to-Peer (P2P) file sharing accounts for a very significant part of the Internet's traffic, affecting the performance of other applications and translating into significant peering costs for ISPs. It has been noticed that, just like WWW traffic, P2P file sharing traffic shows locality properties, which are not exploited by current P2P file sharing protocols.

We propose a peer selection algorithm, Adaptive Search Radius (ASR), where peers exploit locality by only downloading from those other peers which are nearest (in network hops). ASR ensures swarm robustness by dynamically adapting the distance according to file part availability. ASR aims at reducing the Internet's P2P file sharing traffic, while decreasing the download times perceived by users, providing them with an incentive to adopt this algorithm. We believe ASR to be the first locality-aware P2P file sharing system that does not require assistance from ISPs or third parties nor modification to the server infrastructure.

We support our proposal with extensive simulation studies, using the eDonkey/eMule protocol on SSFNet. These show a 19 to 29% decrease in download time and a 27 to 70% reduction in the traffic carried by tier-1 ISPs. ASR is also compared (favourably) with Biased Neighbour Selection (BNS), and traffic shaping. We conclude that ASR and BNS are complementary solutions which provide the highest performance when combined. We evaluated the impact of P2P file sharing traffic on HTTP traffic, showing the benefits on HTTP performance of reducing P2P traffic.

A plan for introducing ASR into eMule clients is also discussed. This will allow a progressive migration to ASR enabled versions of eMule client software.

ASR was also successfully used to download from live Internet swarms, providing significant traffic savings while finishing downloads faster.

1. Introduction

Over recent years Peer-to-Peer (P2P) File Sharing (FS) applications have become very popular. They are used all over the world as an effective way of distributing files to a large population [1]. P2P is particularly well suited to distribute large files due to its inherent scalability: the more peers join the network, the more bandwidth is available.

P2P networks allow software producers and independent artists to distribute content without the need for the large investments in servers and network connectivity required by conventional approaches. Established content producers themselves have begun to warm up to the idea of using P2P technology as a sales channel, instead of regarding it solely as a threat [2]. The commercial endorsement of P2P FS networks will allow them to continue their growth as the dominant Internet traffic generators. Different measurement studies indicate that P2P traffic

☆ This work was partially supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds.

* Corresponding author.
E-mail addresses: ricardo.pereira@inesc-id.pt (R. Lopes Pereira), teresa.vazao@ist.utl.pt (T. Vazão), rodrigo.rodrigues@inesc-id.pt (R. Rodrigues).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.10.014

represents from 35% [3] to 70% [4,5,1] of the traffic carried by Internet Service Providers (ISPs).

ISPs are left at a crossroad. On the one hand, they are confronted with the performance expectations of most customers (who do not use P2P and suffer a performance penalty on their applications), with the bandwidth costs of supporting P2P traffic and with the potential legal issues of P2P, as some of the shared files infringe copyrights. On the other hand, with the revenue potential of clients that use P2P and the pressure to maintain network neutrality. A compromise must be reached between the needs of ISPs, who want to use their networks to provide service to all their customers while minimising their investment, and P2P users, who expect to be able to use P2P applications with reasonable performance.

Current P2P applications do not consider network topology when selecting peers with whom to share files. For instance, a peer may download a file from another peer outside its ISP when that file might be available from peers within the same ISP, incurring higher cost for the ISP. However, several studies have shown that P2P FS swarms show locality properties which are not exploited by current P2P file sharing protocols [6–9]. ISPs will benefit from the use of a peer selection algorithm which takes into account topology information to guide peer selection, favouring local peers.

In this paper we present Adaptive Search Radius (ASR), a peer selection algorithm that allows popular P2P FS applications, such as eDonkey/eMule (herein referred to simply as eMule), BitTorrent or Gnutella, to generate less Internet traffic, while improving the download duration perceived by the users. This algorithm limits the distance, in network hops, of the peers a node may download from, promoting the creation of small, local clusters of peers exchanging a file, instead of large worldwide networks of peers. Each peer still receives the same amount of traffic (it must receive a full copy of each downloaded file), but each data packet travels fewer network hops, and consequently consumes bandwidth over fewer links [10,11].

By fetching data from closer (in network hops) peers, fewer data will cross inter-ISP links, namely expensive international inter-ISP links. This is expected to greatly reduce ISPs' expenses with peering traffic. Peers within the same ISP should be preferred over peers on other ISPs, allowing for a better utilisation of the less expensive intra-ISP links. The freed bandwidth on the inter-ISP links will be available for other applications, which may benefit from better performance by not having to compete with as much P2P traffic.

ASR is a low overhead, easily implementable solution which does not require knowledge of the underlying Internet topology. Using cross-layer information, ASR may be implemented with zero overhead. It may be deployed by gradually upgrading the existing P2P client software [12].

This paper is organised as follows. Section 2 reviews and discusses related work. A background on eMule is provided in Section 3. Adaptive Search Radius is described in detail in Section 4. In Section 5 we present our simulation evaluation, in which we validate ASR against Biased Neighbour Selection (BNS) and traffic shaping using eMule. In this Section we also evaluate the impact of P2P traffic on the performance of HyperText Transfer Protocol (HTTP) and how ASR may alleviate it. Finally, a migration path for the introduction of ASR is proposed and evaluated. Section 7 presents our experience of using an ASR enabled eMule client to download files from live Internet swarms. Section 8 discusses some of the open issues of ASR and the obstacles and opportunities for the introduction of ASR on the Internet. We conclude this paper and present our plans for future work in Section 9.

2. Related work

In the past few years, as the growth of P2P FS traffic became obvious, researchers began working towards solutions to the problem of containing this traffic, resorting to traffic locality. These solutions may be divided into three approaches: the ones where servers/tracker or peers rely on data from the network provided by its operators (ISPs) in the form of oracles; the ones that concentrate the locality efforts on the server/tracker which, by itself, gathers together close-by peers; and the ones which follow the end-to-end architecture of the Internet, having the peers themselves gather information and determine the best peers to use.

2.1. ISP supported solutions

BNS is a locality-aware method for reducing inter-ISP P2P traffic that takes network topology into account [13]. This method, which is applied to BitTorrent, may either be implemented by the tracker with collaboration from the peers or by the ISPs. When implemented on the tracker, peers indicate their ISP, so that the tracker may group them together. When implemented by the ISP, a transparent redirector is used, which, using Deep Packet Inspection, intercepts the communication from peers inside an ISP's network with the tracker. The sources (peers) provided by the tracker are filtered and replaced in order to force the peer to communicate mainly with other peers within the ISP's network. In order to provide worldwide benefits, every ISP would have to install it or every peer and tracker would have to be upgraded.

ISPs have a unique knowledge of their networks which can never be discovered by a client, including information regarding the financial cost associated with each link, peering agreements, routing policies and current utilisation of links. Being that the reduction of the costs supported by the ISPs is the central objective of P2P localisation solutions, it makes sense to use this information. As such, oracle proposals have emerged, where ISPs provide clients with interfaces for discovering the ISPs preferences for routing P2P traffic or for sorting possible peers. P2P clients consult the oracle in order to select which peers, among the ones it knows, should be used [14–16]. Benefits are even greater when ISPs cooperate among themselves.

The IETF has formed a working group entitled Application-Layer Traffic Optimization (ALTO),[1] whose main goal is to design a service that P2P applications can use to achieve better-than-random neighbour peer selection with regard to

[1] http://www.ietf.org/html.charters/alto-charter.html.

the mapping of the overlay topology to the underlying Internet Protocol (IP) network [17].

In summary, ISP-driven oracles enable the most complete decision algorithms as they have access to unique data. However, they face obstacles that may difficult their wide adoption. From the ISP perspective, they may be regarded as an information leak, revealing otherwise private information about its relations with other ISPs. Furthermore, as long as P2P networks are used for the exchange of copyrighted content, ISPs might not wish to associate themselves with, or be in a position to exert control over P2P traffic. For users, who sometimes rely on encrypted P2P protocols in order to protect their privacy, communicating to ISPs which peers they might exchange data with may pose privacy concerns. In order to be implemented, these solutions require the active participation of the ISPs and most also require P2P developers to adapt their applications.

2.2. Server/tracker implemented solution

The use of the tracker to make BitTorrent peers from the same ISPs communicate among each other is proposed in [18]. Peers need not be modified, as the tracker uses preloaded IP address to Autonomous System (AS) mappings to group peers. This technique showed inter-ISP traffic savings in controlled experiments based on data gathered through a crawl of a large number of real swarms, but also revealed that extreme locality may lead to swarm partition.

Pre-loaded IP to AS mappings were also used in [19], this time supplemented with AS relationship maps. This allowed peers and trackers to determine the AS distance between two peers using their IP addresses, with 90% success rate. Also focused on BitTorrent, this work proposed that AS distance be used in three distinct ways: by the tracker, to provide the closest peers; by the peer, to choose which piece to download next; and by the peer, to determine which peers to unchoke.

In summary, server/tracker implemented solutions could have a significant impact if applied to the most popular servers. However, ISPs do not control these servers, and they exist in large numbers, each controlled by different entities. Some solutions also require modifications to the client software. The required combined upgrade effort may be difficult to achieve. A more significant obstacle is that some solutions require Internet topology information, which may be difficult to gather. Also, as clients are provided with few peers outside their ISP, should those depart the P2P swarm, partitions may occur, even if temporary, negatively impacting download performance.

2.3. Client implemented solutions

The Ono plug-in for the Azureus (now Vuze[2]) BitTorrent client aims to reduce inter-ISP traffic. Ono places this burden on the peers, having them prefer to exchange file pieces with nearby peers over the distant ones [20]. In order to determine which other peers are close to themselves, Ono piggybacks on the Akamai[3] Content Distribution Network. Peers which share the same Akamai servers are considered to be in the same network. However, Ono relies on a commercial service which does not profit from the load imposed by it and that may, in the future, become unavailable to Ono. It also requires peers to exchange information about the Akamai servers assigned to each one in order to determine the correlations and thus will only work with other Ono enabled peers. Additionally, another study found that it is only able to reduce inter-ISP traffic when the swarm is large enough, providing little benefit in most cases [21].

A more direct approach is used by Least Hops First (LHF), which exploits locality in BitTorrent. It performs Internet Control Message Protocol (ICMP) ping probes to its peers and determines which ones are closest by reading the Time-To-Live (TTL) value of the replies [22]. LHF works with BitTorrent. It will measure the distance to the 50 peers provided by the tracker and connect to the $35 - k$ closest ones. k other peers will be chosen randomly to complete the 35 connection used by BitTorrent, in order to ensure that the swarm maintains the Small World properties [23]. The use of ping probes adds some overhead and may be impossible to perform when routers/firewalls block ICMP traffic.

TopBT expands the peer unchocking mechanism of BitTorrent by taking AS distance into account [24]. Instead of choosing to upload to the 4 peers from which it downloads the fastest, TopBT sorts peers according to their download/distance ratio, using rounds of several minutes rather than 10 s. This favours close-by peers over distant ones, reducing inter-AS traffic while maintaining a preference for faster peers, which prevents performance from being impacted. The distance metric can be either AS hops, measured using traceroute and IP to AS maps, or IP hops, measured using ping.

In summary, client implemented solutions can be deployed by the application developers, without requiring the redeployment of servers/trackers or the support of the ISP industry. Should these solutions also contribute to improve download performance, we can expect application developers to deploy them, as this will result in a competitive advantage for their applications. However, some of the existing proposals are difficult to implement because they rely on topological information difficult for peers to obtain [19]. Others, such as LHF or TopBT require ICMP, which may be blocked by some ISPs. Ono and [19] are not backwards compatible with other peers which do not implement them, making deployment more complicated. Also, we believe that simply using network distance as the peer selection metric may reduce the robustness of the swarms, as rarest part selection algorithms may be impacted.

3. The eMule protocol

The eMule protocol uses a centralized server, which keeps track of which files each peer shares [25,26]. There may be several servers, but each peer only connects to a

[2] http://www.vuze.com.

[3] http://www.akamai.com.

single one, establishing a Transmission Control Protocol (TCP) connection. When a user wants to download some content, it starts by performing a file query, to which the server responds with the matching files.

To start a download, the peer asks the server for the list of peers sharing the file. A peer may contact other servers, using User Datagram Protocol (UDP), to retrieve more answers to both file and peer searches. While downloading files, the peer periodically (every 15 min) contacts the server in order to learn of more peers sharing the file.

Files are split into parts (of 9.5 MB each) and peers only share full parts. In order to download from another peer (uploader peer), a TCP connection is established. The downloader asks which file parts is the uploader sharing, and determines if the uploader can provide any new file parts. If it can, the downloader asks for a chunk (180 KB) of a file part. It will keep on asking for file chunks as long as the uploader is willing to provide them. When the uploader cannot provide an upload slot to the downloader, it will place the downloader in a queue, together with the other downloaders which are waiting for their turn to download. When queued, the downloader closes the connection and waits for the uploader to connect back, indicating that the download may proceed.

When an uploader receives a request for a file chunk, it determines what to do according to the downloader's score and who else asked for file chunks. Each peer, in its uploader role, assigns a score to the peers (downloaders) which have asked for a file chunk. This score is proportional to the elapsed time since the downloader's first chunk request and the amount of data the peer has received from this downloader. It is also inversely proportional to the amount of data already sent to the downloader.

Each peer provides only a few download slots. The peers at the top of the queue are allowed to use one of the slots. All the others have to wait until they reach one of the top positions. In order to avoid fluctuations, once a peer is allowed to download, its score is doubled during 15 min. If the peer who is to start downloading is no longer connected, a callback connection is performed. Each peer tries to download from all the other peers it knows about, as such, the upload queues are usually very long. In order to keep track of the parts being shared, downloading peers will periodically (every 20 m) contact the peers which have queued them, using UDP.

4. Adaptive Search Radius

We designed ASR as a peer selection algorithm to be used with eMule and other P2P FS applications. The ASR algorithm is run locally and independently by each peer. Its goals are to significantly reduce the amount of inter-ISP and global Internet traffic due to P2P FS, while having no impact on the download performance perceived by users. This is accomplished by having each peer restrict the set of neighbours it will download from.

4.1. Algorithm description

The basic principle behind the algorithm is that instead of downloading from all other peers it knows, the client will restrain from downloading from those peers which are topologically distant. To achieve this, ASR defines a *search radius*, measured in network hops and centered on the client, which encompasses the subset of peers from which it may download. Defining this search radius involves a trade-off that must be carefully managed. On the one hand, the shorter the search radius, the fewer links will be used to transport the traffic. But on the other hand, there is the need to ensure that peers maintain connections with a sufficient number of other peers such that the necessary file parts can be obtained, avoiding having to frequently discover new peers. To capture this trade-off, we continuously measure the availability of each file part among the peers that a node is connected to, which allows us to set the number of network hops of the search radius as a function of this metric. In particular, ASR attempts to use the shortest search radius that guarantees a minimum availability for every file part.

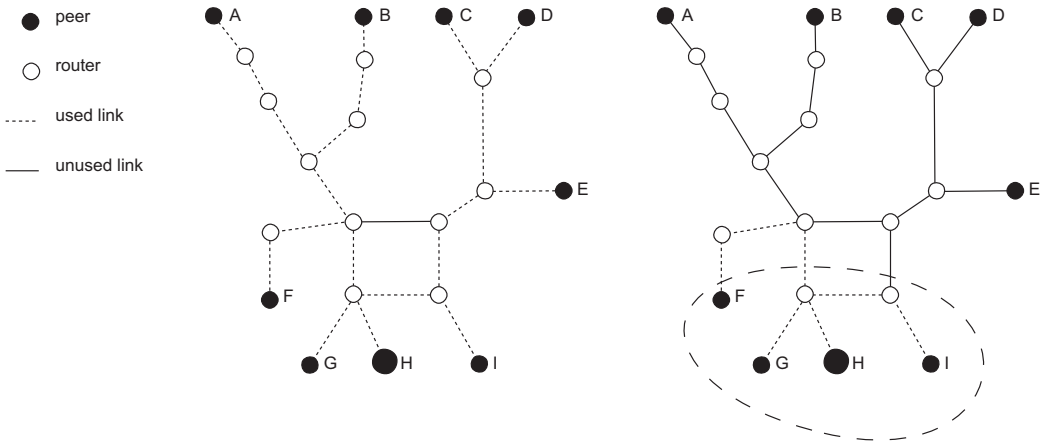
Fig. 1 depicts the behaviour of a P2P file sharing client (node H) while downloading a file. In Fig. 1(a), peer H does not use ASR to filter out distant peers, downloading from all of them. Therefore, download traffic crosses most links, contributing to their congestion.

Fig. 1(b) shows what happens when ASR is used. In this example, it is assumed that ASR calculated a search radius of 4 network hops. This means, that peers F (4 hops), G (2 hops) and I (3 hops), together provide the required minimum number of sources for all the pieces peer H still needs to download. Since ASR peers restrain from downloading from peers outside the search radius, information travels fewer network hops, releasing capacity on all other links. Peer H still needs to download a full copy of the file, but the download is performed from close-by peers, impacting fewer Internet links.

An important downside of restricting the search radius is that a peer will issue requests for file parts to fewer peers than it would in case there were no restrictions on who to contact. This might appear to affect the download time, particularly in the case of eMule, given that its clients discover several hundred peers that share the file and try to download from all of them. Intuitively, the more requests the client issues, the faster the download. However, since all downloading peers compete for the upload slots and ASR will cause upload queues to become shorter, peers will wait less to start downloading.

As we mentioned, we need to dynamically adjust the search radius according to the diversity of file parts that are held by the set of peers that a node is connected to. For this purpose, we define a metric called *file availability*. This is defined as the number of peers, within the current search radius, sharing the rarest file part that the peer has not downloaded yet. In other words, it is the minimum number of copies, available within the search radius, for every file part yet to be downloaded. As uploaders are contacted, their distance (in network hops) and the file parts they share are determined. Fig. 2 shows how file availability may be calculated for a given file and search radius (*numHops*).

Initially, ASR peers contact every peer they learnt about, so we set the search radius to a value that is large enough to encompass the entire Internet (e.g. 64 hops). After



(a) Without ASR (b) With ASR using search radius of 4

Fig. 1. Links crossed by file download traffic by node H.

learning (or updating) which file parts a peer shares, or when a peer leaves, the file availability metric is calculated. Also, when new peers are discovered, they must be contacted so that their distance and shared file parts are learnt for the search radius to be recomputed. If file availability is larger than the maximum threshold, the search radius is reduced by one hop while file availability remains larger than the minimum threshold. If all peers within the search radius have been contacted but file availability remains below the minimum threshold, the search radius is increased by one hop. Each peer continuously calculates the search radius for each of the files it is downloading. Fig. 3 shows how the search radius is calculated. The algorithm uses different constant values for the minimum (*MinAvailability*) and maximum (*MaxAvailability*) part availability thresholds. *MinAvailability* defines the minimum number of sources for each file copy. *MaxAvailability* limits the number of possible peers so that locality is exploited.

The condition that all peers have been contacted (line 5 of Fig. 3) ensures that the information about the file parts shared by the peers within the search radius is up-to-date before it is expanded.

ASR only limits the distance to the peers that a peer can download from, not the distance to peers it can upload to, since this would hinder the capability of a solitary peer (or set of peers) to begin downloading a file.

We've chosen IP hop distance as the metric to define the search radius. This simple metric can be determined by the client, without adding any overhead and is directly related to the network topology, which we intended to exploit in order to reduce inter-ISP traffic. Recent work has also found this to be the best metric [21].

```
FileAvailability(file, numHops)
 1  // count available sources for each file part
 2  for peer in GetSources(file)
 3       do // only consider peers within the specified distance
 4          if PeerDistance(peer) ≤ numHops
 5            then partsAvailable ← SharedParts(peer, file)
 6                 for x ← 1 to NumParts(file)
 7                      do partCopies[x] ← partCopies[x] + partsAvailable[x]
 8  // determine availability of rarest part yet to download
 9  minimumAvailability ← MAX-INT
10  for x ← 1 to NumParts(file)
11       do if not PartDownloaded(x, file) and partsAvailable[x] < minimumAvailability
12            then minimumAvailability ← partsAvailable[x]
13  return minimumAvailability
```

Fig. 2. Determining the availability of the rarest part.

```
CalculateSearchRadius(file, currentSR)
1  currentAvailability ← FileAvailability(file, currentSR)
2  if currentAvailability > MaxAvailability
3     then while FileAvailability(file, currentSR−1) ≥ MinAvailability
4               do currentSR ← currentSR−1
5  elseif currentAvailability < MinAvailability and AllPeersContacted(file, currentSR)
6     then currentSR ← currentSR+1
7  return currentSR
```

Fig. 3. Calculating the search radius.

4.2. Determining the number of hops to a peer

Determining the distance to other peers is fundamental for using ASR. We suggest using two methods to achieve this, both of which have very little or even no overhead.

The first one is to perform a UDP ping, as shown in Fig. 4, where peer A determines its distance to peer B by sending an UDP packet to an unused port (UDP ping) on the other peer. Peer B responds with an ICMP Destination Unreachable (Port Unreachable) message. The difference between the TTL value with which the ping packet leaves peer A and the TTL value that reaches peer B (returned in the ICMP message) is the distance to the peer.

As ICMP messages may be blocked by firewalls and pinging introduces an extra Round-Trip-Time (RTT), protocol generated data packets might also be used to measure the distance. If one knows the TTL value of the packets leaving the remote peer, the distance can be determined by accessing the TTL field of the arriving packets. Since the TTL value of a packet leaving a host varies from one operating system to another, this value could be transmitted via an extension in the P2P FS protocol or inferred though the peer identification. As dominant modern operating systems (Windows, OS X, GNU/Linux) use either 64 or 128 as the default TTL value [27], an alternative approach is to use Eq. (1). This method will provide reasonable accuracy as Internet path length is not expected to exceed 64 hops (GNU/Linux traceroute uses a default maximum path length of 30 hops).

$$hop\_distance = \begin{cases} 64 - ttl + 1, & \text{if } ttl \leqslant 64, \\ 128 - ttl + 1, & \text{if } ttl > 64. \end{cases} \quad (1)$$

This mechanism is depicted in Fig. 5. Peer B knows that the initial TTL of packets sent by A is 64. As packets arrive at B with a TTL of 62, B learns that A is 3 hops away. Peer A receives packets from B carrying a TTL of 126 and knowing that B's initial TTL is 128, can determine it to be 3 hops away. This second mechanism has no overhead. Should the distance be determined by monitoring the TCP SYN packets, it will be available after the TCP connection establishment phase, so no protocol messages need be spent on distant peers. However, this method requires either the use of UDP (which is possible with eMule) or a cross layer approach, as the standard Socket API only provides applications with access to the TTL field of UDP messages. One possible cross-layer technique would be to use a packet capture mechanism, such as the libpcap library used by tcpdump, to read TCP SYN packets [28]. We've successfully tested this solution in a prototype.

4.3. ASR interaction with eMule

The design features of ASR might interfere with some of the mechanisms used by the P2P FS protocol. In the case of

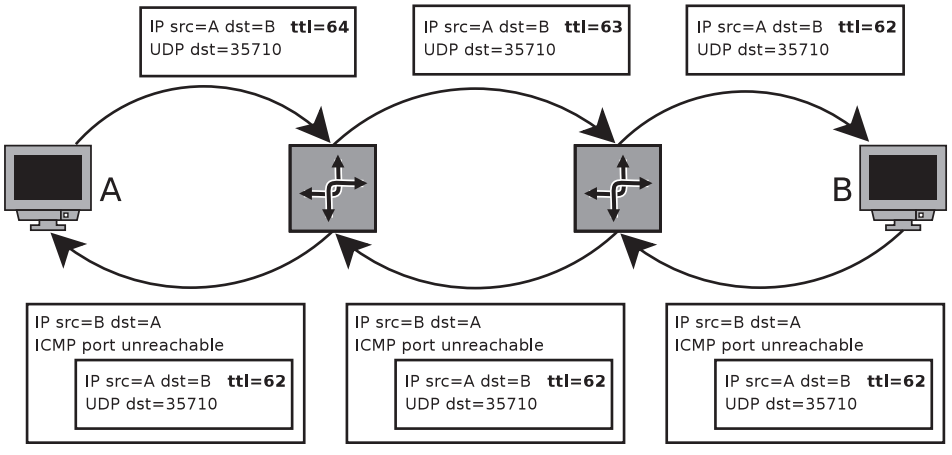


Fig. 4. Hop distance measurement using UDP ping.

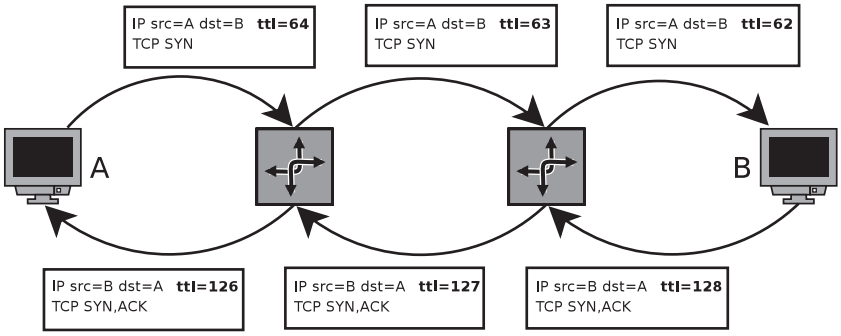


Fig. 5. Cross-layer hop distance measurement using TCP.

eMule, this issue arises in the context of the mechanism that regulates downloads, which is the download queue and the score system.

Consider what happens when a peer joins the swarm. Such a peer will usually have to wait a long time before it is able to download for two reasons: it has nothing to upload to the other peers and it has waited little time. As such, its initial score will be very low. Only after some time, which will depend on the number of competing peers in the queues, will the peer achieve a high enough score to be allowed to download. As the peer enlists in the download queue of every peer it knows about, eventually it will be allowed to download from several sources simultaneously. Once the peer has downloaded a full piece, it will be able to upload to others, thus increasing its score and its chances of being allowed to download.

ASR will interfere with this mechanism by limiting the number of peers each peer will connect to. However, this is not problematic since the decreased number of peers to download from is compensated by having to wait less to start downloading, as upload queues are shorter. Since each downloading peer will restrict the set of peers it downloads from, each uploader will have to satisfy fewer requests. Besides, when the search radius is reduced, an eMule client need not immediately remove itself from the download queues of the nodes which fall outside the search radius. Instead, it may wait for a callback, and only them reject the download slot. This will ensure that if the search radius is again increased after having shrunk, the client will not have to again ask to be placed at the bottom of the download queue.

Another issue is related to the fact that eMule's score system strives to achieve reciprocity over a long time span. As the search radius is expected to decrease over time, at least for popular files, some of this reciprocity may be lost as some of the peers with which data was exchanged are left outside the search radius and the peer will not be able to take advantage of earned credits. However, as both given and received "download credits" are expected to be lost, the overall impact should be small.

5. Simulation study

In this section we first evaluate the performance of ASR against eMule's. We then compare both to BNS and traffic shaping. This section ends with an evaluation of the impact of P2P on HTTP traffic.

The impact of ASR on the eMule protocol was evaluated through simulation using the SSFNet 2.0[4] network simulator which provides simulation of the network, transport and application layers [29]. Since at the time of our initial work there was no P2P FS protocol implementation for a network simulator, we implemented the eMule protocol on SSFNet.[5] We followed all the relevant protocol characteristics, leaving out features such as file compression, integrity verification (file part hashes) and secure authentication. As a single server was used in the simulations, there was no need for peers to exchange file sources. All communication among peers is performed using TCP, even though eMule allows UDP to be used for refreshing information about the file parts shared by other peers. We do not believe this simplification to have a significant impact on the simulation results as peers are only contacted once every 15 min, making TCP overhead negligible.

After implementing and evaluating ASR against regular eMule, we implemented BNS (the first ISP-assisted proposal) and traffic shaping (used by some ISPs to reduce P2P traffic) to benchmark ASR against these alternatives.

Although BNS was originally proposed for use with BitTorrent, we adapted it to work with eMule. We believe that biasing the set of peers provided to each peer so that it is mainly comprised of peers from the same ISP is equally applicable to both protocols. We implemented BNS by modifying the eMule server: whenever a peer asked for peers sharing a certain file, from the 50 peers provided by the server, up to 40 would be within the same ISP.

Traffic shaping is sometimes used to limit the impact of P2P traffic and affects P2P performance and behaviour. We chose to implement a bit bucket, as this simple mechanism was sufficient to illustrate the impact of traffic shaping. The bit bucket is replenished every 100 ms and its maximum capacity corresponded to 1s of the transmission rate of the bit bucket. The bit bucket rate varied for the different scenarios in which it was used.

[4] http://www.ssfnet.org.

[5] The software we developed is available for download at http://cnm.tagus.inesc-id.pt/project/adaptive-search-radius.

5.1. Simulation scenarios – common characteristics

Evaluation of ASR was performed with varying network topologies in order to avoid any possible bias introduced by a particular topology. We simulated both the overlay and the transport and network layer, yet we used a number of peers similar to that found in other simulation studies [13]. The results presented in this article were produced using 5 different network topologies. These are transit-stub topologies generated using the GT-ITM topology generator [30]. Here, transit networks represent tier-1 ISPs, being used to connect stub networks. Stub networks represent tier-2 ISPs, which are national or regional ISPs, providing service to end-users. We extended the original network graph created by GT-ITM with hosts, by considering the original graph to contain only routers and connecting a few hosts to each stub router. These point-to-point connections represent Asymmetric Digital Subscriber Line (ADSL) connections from ISPs to their clients.

Table 1 summarises the main characteristics of the topologies that were used. For instance, network ts450 is the largest one. It has 3 transit networks. Each transit router provides connectivity to 3 stub networks. There are 56 stub networks. Twenty extra links were added between randomly chosen stub networks in order to simulate direct connection between local ISPs. Another twenty links were randomly used to connect transit and stub networks, simulating alternative connections for local ISPs. In total, the network is comprised of 450 routers. To each of the tier-2 routers, one or two hosts were connected, totalling 647. Each host was used to run an eMule peer. The maximum distance between two nodes (routers or hosts) is 15 hops, and the average distance between routers is 10.2 hops.

In all networks, hosts are connected to their routers using 0.5 or 1 Mb/s links. Stub routers are connected among themselves using 5 or 10 Mb/s links. Stub networks are connected to each other and to transit networks using 2 or 5 Mb/s links. Transit routers are connected using 10, 20 or 50 Mb/s links. Transit networks are connected using 10 or 20 Mb/s links. While these are not realistic rates, their intent is to represent realistic relative rates, at approximated orders of magnitude, while making simulations feasible on affordable hardware.

The eMule server was run on one of the transit nodes, in order to be approximately equidistant to every peer. This was not expected to have any significant impact on routing performance. ASR was configured to increase its search radius when the file availability was below 3 and to reduce it when file availability was greater than 6. These values were found to provide a reasonable tradeoff between download time and proximity exploitation for all our test scenarios.

Simulations were repeated with different initial seeds and the average values are displayed in the tables. We obtained confidence intervals of 5% for a confidence level of 95%.

5.2. Performance of ASR

We evaluated the performance of an ASR enhanced version of eMule against that of the original eMule protocol using the ts450, ts294-2 and ts171 topologies. These tests were performed using a large scale model, similar to what happens in the Internet, where several files are being shared at the same time. Each simulation was run for a day (in simulated time). Table 2 presents the main simulation configuration parameters.

Peers do not all start downloading the first file at the same time, in order to distribute load on the server. From time to time, each peers starts a new download. After finishing a download, the peer will continue to share it for same time, as a seeder. As such, a peer could be downloading several files and seeding several others at once. A thousand files were available for download. In order to perform a download, a peer would select one file according to their popularity [31]. Files were regularly replaced, one every time period. When a new file was introduced, all files less popular than the new file were shifted one position down the popularity rank and the least popular file was removed. Each file was shared by some peers (seeders) during the entire simulation: an average of 3.88 seeders per file for ts450, 1.68 for ts294-2 and 2.12 for ts171. Each peer provided 10 upload slots and maintained a maximum of 50 active connections to other peers.

Every network was simulated under light and heavy load. The light load was accomplished by having each peer start a new download every 6 h, while doing it every 2 h generated the heavy load. The heavy load scenario is an extreme one, as the rate of download request will always be larger than the rate of download completion. In the real world users would eventually give up on some of the downloads, lowering the load.

In both cases (eMule and ASR) the workload was the exact same: the same peers started downloading the same files at the same time.

5.2.1. Analysis of the results

We will examine in detail the result of the light load simulation using the ts450 network, then we will compare the different simulation scenarios.

Table 1
Simulated networks.

Name	Transit networks	Stub networks	Routers	Peers	Diameter	Average distance between routers
ts450	3	56	450	647	15	10.2
ts350	2	20	350	840	15	10.1
ts294–1	2	28	294	560	18	12.1
ts294–2	2	28	294	408	18	12.1
ts171	1	18	171	240	16	10.1

Table 2
Simulation parameters.

Behaviour	Distribution	Parameters
Total files shared	Constant	1000
First and inter-download delay (light load) (s)	Uniform	*min* = 5, *max* = 43.200
First and inter-download delay (heavy load) (s)	Uniform	*min* = 5, *max* = 14.400
Seeding time (s)	Uniform	*min* = 5, *max* = 43.200
File popularity (which file to download)	Zipf	$p^k = 1$, $\rho = 1.0$
Shared files replacement period (s)	Uniform	*min* = 300, *max* = 3.600
New file popularity rank	Zipf	$p^k = 1$, $\rho = 1.0$
File size (MB)	Uniform	*min* = 3, *max* = 900
Number of upload slots per peer	Constant	10

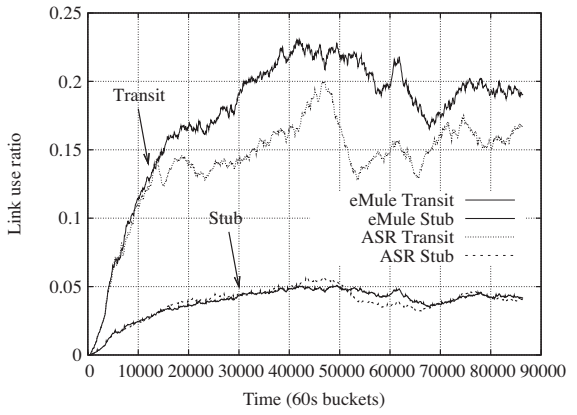


Fig. 6. Link utilisation.

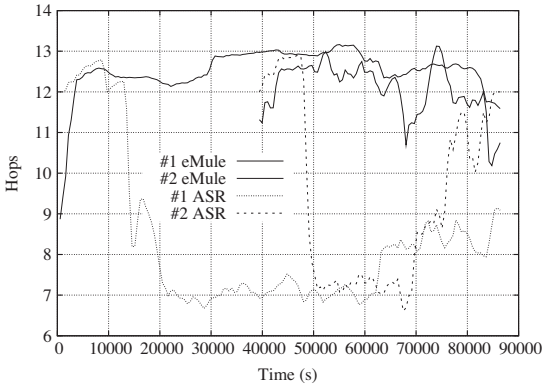


Fig. 7. Search radius for the two most downloaded files.

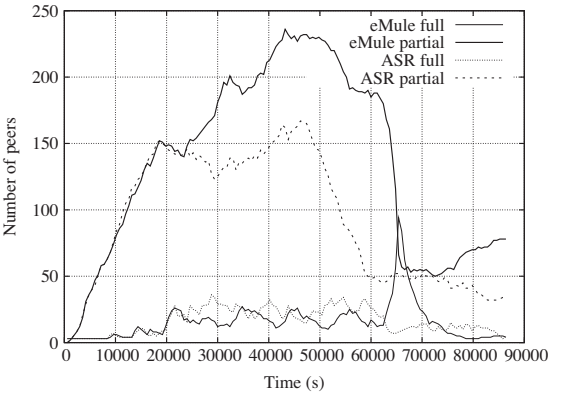


Fig. 8. Peers sharing the most downloaded file.

The primary goal of ASR is to reduce the impact of P2P traffic on the Internet. We separately measured the traffic in transit and stub networks, adding the traffic that crosses every link of each type. Fig. 6 shows the evolution of each type of traffic using the original eMule protocol and the ASR-enabled version. We can see that while there is no significant difference when it comes to stub traffic, ASR leads to noticeable traffic reductions in the transit networks, showing that packets travel fewer hops in the transit networks.

Fig. 7 shows that, as expected, the traffic reductions using ASR stem from limiting the distance to the peers contacted. The Figure shows the evolution of the average search radius used by the peers downloading the two most popular files (#1 with 557 downloads and #2 with 264). It also shows the average distance to the peer that is furthest away, which is the search radius effectively used by eMule. Over many simulation runs, the most downloaded files will also be the ones which contribute the most to Internet traffic/congestion (a uniform file size distribution was used). As soon as enough peers have downloaded some file parts, file availability rises, allowing ASR peers to lower their search radius, abandoning the upload queue of the most distant peers. Using only the closest peers reduces the number of links used, and an immediate reduction on transit link traffic can be observed around 15,000s, when peers began to reduce the search radius for the most popular file. Another reduction in traffic can be observed at around 50,000s coinciding with the decrease of the average search radius for the second most popular file. When not enough peers are found to satisfy the minimum file availability, the search radius increases without bounds.

ASR managed to reduce the search radius for the most popular files, down to those which were downloaded by only 29 peers. Overall, ASR resulted in traffic savings for 77% of the downloads.

The bandwidth released on some links by one peer may be used by a different peer to download faster. Fig. 8 shows the evolution of the swarm for the most popular file. This information was gathered from the eMule server, were every peer is registered. For both ASR and eMule, the number of peers fully and partially sharing the file are plotted. We can observe that the number of peers sharing partial content starts by rising, as more peers start downloading, but as ASR allows downloads to finish faster, this number decreases faster using ASR than eMule. Peers only share the file for a while after finishing the download, as such, the number of peers sharing the full file is highest when

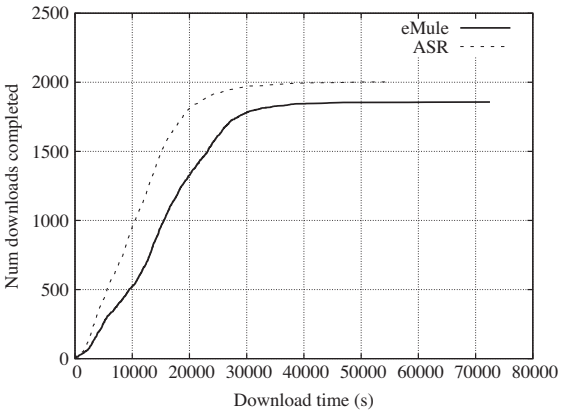


Fig. 9. Download times.

many peers finish their downloads. This is observable for eMule around 65,000s where we see a peak of peers sharing the full file after the number of peers still downloading decreased rapidly. A large number of peers sharing the full file is a good opportunity for others to finish their download as these provide sources for the full file while not competing for the upload slots.

ASR would never be adopted by users should it inflict noticeably longer download times. Fig. 9 shows the download times achieved by ASR and eMule. It shows that ASR not only finished the downloads faster, but was also able to finish downloading a few more files. This happens because the bandwidth released on some links by one peer may be used by a different peer to download faster.

Table 3 summarizes the results of the different simulations that we performed. In order to facilitate reading, we present only the performance of ASR relative to eMule instead of the absolute values.

We can observe that ASR is always more efficient, especially under heavy load. It finishes more downloads using less transit traffic and, in the light load scenarios, less stub traffic. Going from the smaller network (ts171) to the larger (ts450), we observe that the benefit of using ASR increases as the network size grows. In this scenario, the bigger the network, the larger the number of hosts downloading the same files.

5.3. ASR vs BNS vs traffic shapping

We compared the performance of ASR with another technique proposed in the literature (BNS) and a common ISP practice (traffic shaping). As a baseline for our evaluation we used eMule. We used the ts350 topology, our second largest topology. The larger the network, the closer it would resemble the Internet. However, ts450 had proved too complex for us to be able to run the large number of simulation planned in reasonable time.

We simulated the distribution of a 100 MB file, seeded by 7 peers to the other 833. This depicts a situation where a company uses P2P FS to distribute some content (for instance e-learning) to their various offices, or a flash crowd where hosts continue to share after finishing their downloads. Each simulation is run until all downloads are completed.

Traffic shaping was used to limit the aggregate P2P FS bandwidth permitted on the stub-stub and stub-transit links, using a bit-bucket. These links represent the connections between ISPs, where costs are more significant, being therefore the ones where ISPs would throttle P2P traffic. Intra-domain links are cheaper to provision. We ran experiments limiting the available P2P FS bandwidth to 4, 3, 2, 1 and 0.5 Mb/s.

We have also decided to evaluate the combination of ASR and BNS, given that these are not conflicting technologies. The use of BNS will allow ASR to converge on nearby peers more quickly, as the peers it learns about will be mostly within the same ISP. From the point of view of BNS, the use of ASR will allow the peer to choose the closest peers from within the ISP, instead of using a random set. Also, while BNS is unable to provide traffic savings when there are no sources within the same ISP, ASR will download from the closest peers.

In order to assess the performance of the various solutions we introduced two new metrics: average number of hops crossed and efficiency.

The average number of hops crossed by each packet was calculated using Eq. (2). It divides the total network traffic by the amount of application level traffic generated by each peer and by the server. The real average number of hops crossed is lower than the calculated value as it does not include any TCP/IP overhead.

$$avg\_hops\_crossed = \frac{stub\_traffic + transit\_traffic}{\sum_{i=1}^{num\_peers} tx\_data\_traffic_i + tx\_data\_traffic\_server} \quad (2)$$

The efficiency of each algorithm was calculated using Eq. (3). To calculate it, we used a simplified scenario, without peer departures, which allowed us to calculate the optimal path for distributing the file, i.e., the one which requires the file to be transmitted across the fewest links.

Table 3
ASR performance gains.

	ts450		ts294-2		ts171	
	Light (%)	Heavy (%)	Light (%)	Heavy (%)	Light (%)	Heavy (%)
Finished downloads	107.8	198.7	100.6	123.3	105.9	114.4
Completed files size	110.2	190.0	101.0	123.4	107.5	114.6
Average download time	73.5	103.6	97.0	103.0	94.6	100.5
Transit traffic	80.1	85.0	91.9	92.7	88.7	96.1
Stub traffic	99.1	118.1	97.9	111.1	98.1	111.5
Num. TCP connections	27.1	66.7	79.1	86.4	63.7	92.7

Table 4
Comparing the different algorithms.

	eMule	ASR	BNS	ASR/BNS
Avg. download time (s)	1552	1439	1434	1389
Transit traffic (GB)	264	160	165	77
Stub traffic (GB)	553	452	502	387
Inter-ISP traffic (GB)	170	115	106	53
Connections (×1000)	150	81	124	72
Avg. hops crossed	9.76	7.31	7.98	5.55
Efficiency (%)	24.57	32.80	30.06	43.25

$$efficiency = \frac{stub\_traffic + transit\_traffic}{filesize \times optimal\_path\_length}, \quad (3)$$

where the optimal path length was determined by using a modified version of the Dijkstra shortest path algorithm. Note that it would be impossible for any protocol to reach 100% efficiency as this does not take into account any of the overheads suffered by the protocols (IP, TCP, eMule).

5.3.1. ASR vs BNS analysis

Table 4 shows the main metrics gathered from the experiments. We can observe that both ASR, BNS and their combination provide significant improvements on all metrics over the use of plain eMule, both from the point of view of the ISPs and of the subscriber. We can observe the difference in the objectives of ASR and BNS in the results. BNS provides a greater reduction in the inter-ISP traffic as it was designed to contain P2P traffic within the ISP. ASR, on the other hand, being designed to decrease the impact of P2P not only on an ISP but on the global Internet, achieves superior results on the reduction of the stub and, more importantly, on the transit traffic.

Also significant is the reduction in terms of number of connections necessary to distribute the file. Here we see that ASR has an advantage over BNS, but not over their combined use. Note that since TCP is an end-to-end protocol, the number of connections affects primarily the performance perceived by the peers. But ISPs may also benefit from having fewer connections to keep track of, which allows for better scaling of any deep packet inspection equipment used by complementary techniques.

We can also observe that the average number of network hops crossed by each P2P FS data packet is reduced by the use of BNS and even more by the use of ASR. Their combined use yields the best results.

The efficiency metric allows us to evaluate how close to the ideal solution each algorithm stands. Once again we can observe that the use of ASR or BNS is advantageous, with ASR having an edge over BNS. Their combination is, once again, very beneficial.

5.3.2. Analysis of traffic shaping impact

Fig. 10 shows the evolution of the amount of inter-ISP traffic used when the allowed inter-ISP P2P rate is varied by traffic shaping. We can observe that the reduction only becomes significant when the bandwidth is reduced to 1 Mb/s. Reducing the bandwidth from 1 to 0.5 Mb/s results in a faster decrease in traffic consumption. The different techniques maintain their relative positions with different bandwidth values, except for 0.5 Mb/s, where ASR provides much greater savings than BNS, indicating that ASR copes better with traffic shaping.

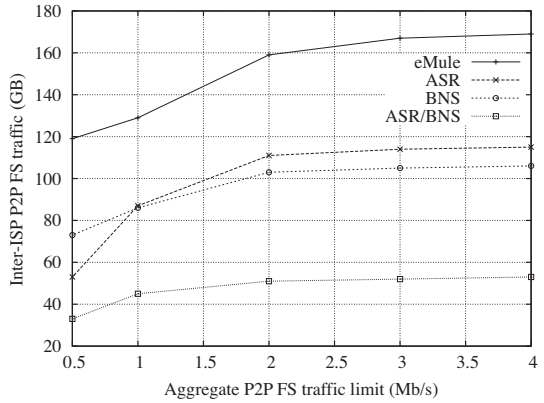


Fig. 10. Amount of traffic exchanged among ISPs versus allowed bandwidth.

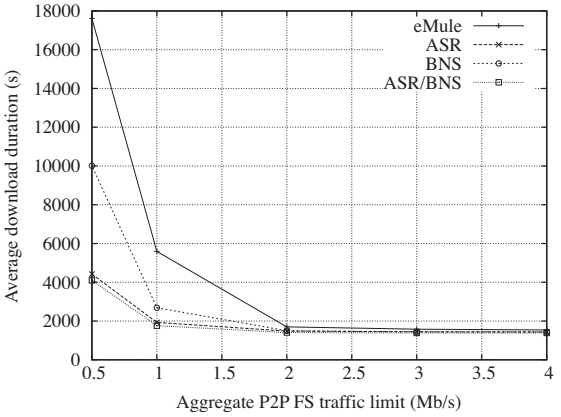


Fig. 11. Behaviour of download time with different bottlenecks.

Fig. 11 shows the behaviour of the average download time achieved by the various techniques under different traffic shaping restrictions. It is noticeable that users pay a high price for the use of traffic shaping. With the plain eMule protocol, it is difficult for ISPs to reduce P2P FS traffic below 2 Mb/s, since their customers will notice a substantial slowdown in their download speeds. However, traffic saving were only significant below this mark. BNS shows a much better behavior, but it also causes clients to observe substantial download slowdown at 0.5 Mb/s, and even at 1 Mb/s. ASR shows a behaviour very close to that of ASR and BNS combined, and both of these alternatives would allow P2P FS traffic to be limited to 1 Mb/s without causing major annoyance to subscribers.

Fig. 12 shows the evolution of the traffic efficiency of the various techniques under different traffic shaping restraints. The relative positions are always maintained.

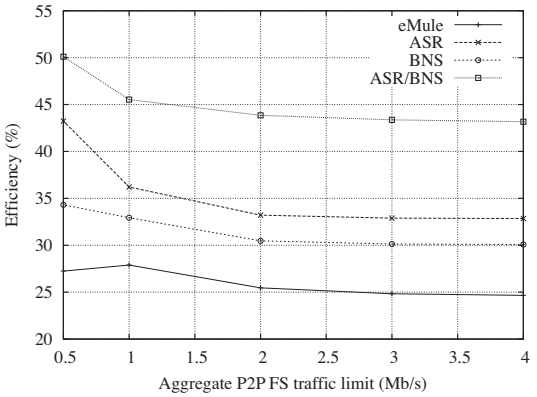


Fig. 12. Efficiency of the various techniques under different bottlenecks.

However, we can see that ASR combined with BNS and, especially, ASR by itself, react better to severe bandwidth restrictions, increasing their efficiency. Using 0.5 Mb/s, the plain eMule protocol reduced its performance, as the decreased bandwidth becomes insufficient for both data and control messages, reducing the capacity of peers to discover new, closer-by peers.

5.4. Impact of P2P traffic on HTTP performance

The large amounts of data exchanged by P2P FS applications contribute to congestion on Internet links. This should have an impact on other applications, which are forced to compete for the available capacity on each link. HTTP is the application that generates the second highest amount of traffic, accounting for 16 to 34% of the Internet traffic in different parts of the world [1]. HTTP is used by more people than P2P FS applications. In order to evaluate the impact of P2P traffic on the performance of HTTP traffic, we added the latter to our simulation scenarios.

The work presented in this section was performed using the same network (ts350) and P2P settings used in the previous subsection. HTTP 1.1 with persistent connections and no pipelining was used. HTTP clients are simulated to perform like real persons, using a model loosely based on the one proposed in [32], which was augmented to introduce different sizes for HTML and binary objects. The HTTP traffic is characterized by the statistical distributions presented in Table 5. A total of 73 HTTP servers were used, 13 placed on end nodes and 60 on stub routers. All P2P peers, either seeders or leachers were used as HTTP clients.

5.4.1. Analysis of the results

We started by analysing the HTTP traffic by itself, having no P2P traffic present. The results are shown in Table 6.

When the experiment was repeated adding eMule traffic, there was a significant penalty to HTTP's performance. The impact on eMule was much smaller, as can be seen by comparison with the results presented in Table 4. While the average download time for HTTP increased 74%, that of eMule only increased 18%. Further, this slowdown occurs in spite of HTTP only being able to conclude 58% of the requests and transfer 56% of the data it had been able to when it had the network to itself. We can conclude that when eMule and HTTP compete for network resources, eMule gets the better part of the resources. In this simulation scenario eMule traffic represented 68% of all the traffic, which is consistent with the upper bound of the current estimates for the Internet [4,1].

Table 5
HTTP behaviour modulation.

Behaviour	Distribution	Parameters
Inter session time	Exponential	$\Lambda = 10$
Pages per session	Normal	$\mu = 3.86$, $\sigma^2 = 2.465$
Page read time	Pareto	$k = 0.1$, $\alpha = 8$
Objects per page	Pareto	$k = 2$, $\alpha = 1.245$
HTML object size	Log-normal	$\mu = 7.63$, $\sigma = 1.001$
Object size	Log-normal	$\mu = 8.215$, $\sigma = 1.46$

We expected the traffic savings afforded by ASR or BNS to have a beneficial impact on the performance of HTTP. In fact, the use of ASR not only improved the performance of P2P FS but that of HTTP as well. For P2P the average download time only suffered an increase of 10% when compared to the scenario where only ASR traffic was present (Table 4). This means that the advantages of ASR over eMule, from the perspective of P2P users, increase when P2P traffic competes with HTTP, as happens on the Internet.

The increased download time allowed ASR to work longer, when compared to the scenario without HTTP traffic, thus increasing its efficiency. The nature of HTTP traffic, where data is mainly exchanged with servers outside the ISP, causes HTTP traffic to compete with P2P traffic for the inter-ISP links and not so much for the intra-ISP ones as there is little HTTP traffic among end nodes in the same ISP. As such, the reciprocity mechanism of eMule will guide peers to other peers within the same ISP where higher data rates can be obtained. This is why the efficiency of P2P transfers increased when compared to the scenario with no HTTP traffic of Table 4.

6. Migration

ASR's main goal is to reduce the impact of P2P traffic on the Internet. But even if this goal is fully achieved, it will not be enough to motivate users to migrate to an ASR enabled implementation of eMule. There must be a more immediate reward: the capacity to download faster. However, we found that without a migration plan, ASR only provides its users with the fastest downloads when it is utilized by 100% of all the P2P users. Traffic savings, on the other hand, are always observed, even without a migration plan.

This section analyzes this problem and proposes a solution that makes ASR attractive in all stages of the deployment.

6.1. Partial deployment

We simulated the distribution of a 200 MB file, seeded by 6 hosts, to 554 other hosts using the ts294-1 network. After completing the download, the peer would continue

Table 6
Impact of P2P traffic on HTTP performance.

		No P2P	eMule	ASR	BNS	ASR/BNS
P2P	Average download time (s)		1836	1592	1665	1488
	Inter-ISP traffic (GB)		161	106	98	48
	Average hops crossed		9.38	7.03	7.65	5.34
	Efficiency (%)		25.54	34.12	31.35	44.89
HTTP	Successful requests (×1000)	1540	886	1150	1120	1300
	Average download time (s)	8.87	15.4	11.9	12.24	10.45
	Downloaded data (GB)	45	25	33	32	38
	Fraction of P2P traffic (%)	0	68	55	58	45
	Total inter-ISP traffic (GB)	149	246	216	206	173

Table 7
Partial deployment using original ASR algorithm.

	eMule	10% ASR	90% eMule	50% ASR	50% eMule	90% ASR	10% eMule	ASR
Avg. download time (s)	4400	5807	4114	4627	3150	3262	2596	3136
Avg. search radius		7.24		7.2		7.29		7.4
Avg. download time (s)	4400	4284		3891		3196		3136
Transit traffic (GB)	458	420		307		169		137
Stub traffic (GB)	677	658		604		529		510
Inter-ISP traffic (GB)	221	206		165		111		99
Num. connections (×1000)	747	631		313		114		102
Avg. hops crossed	10.24	9.73		8.22		6.31		5.84

to share the file for an average of 2000 s (exponential distribution, $\lambda = 0.0005$). Peer download start time was uniformly distributed over the first 4 h of simulation ($min = 0$, $max = 14400$).

The goal of this experiment was to observe what happens as more and more users migrate to an ASR-enabled implementation of eMule. We ran experiments with 5 different scenarios: every peer using regular eMule, every peer using ASR enabled eMule, half the peers using each method, 90% using one method and the rest using the other. Table 7 summarizes the simulation results.

The average download time is reduced by the introduction of more ASR peers. The same happens, as expected, to all the traffic types. We can see that the more ASR peers are deployed, the closer data is fetched from. The number of connections between peers (protocol overhead) also declines as the number of ASR peers increases.

From the perspective of an ISP this makes all of the tested partial deployments of ASR benefitial in comparison to the current situation. However, in all partial deployment scenarios, peers using regular eMule experience faster downloads than those using ASR. Users would never consider using a slower application when the one they already use works better. Furthermore, the introduction of ASR peers benefits eMule users, who experience ever faster downloads as the number of ASR peers increases.

The shorter average download times and reduced traffic are explained by the behaviour of ASR peers, who only download from nearby peers. As a consequence, less capacity is used on each link and the upload queues on each peer are shorter. On the other hand, eMule peers try to download from every peer they know. As there is more link capacity available and shorter upload queues they will experience faster downloads. When eMule peers are present, though, upload queues become large, and ASR peers have to wait for their turn to download from only a few peers while eMule peers wait to download from a large set of peers.

6.2. Migration plan

To devise a migration plan which is attractive at all stages, we observe that the key to ASR's success are small upload queues. ASR peers only contact peers within their search radius and expect them to have small upload queues, which does not happen in the presence of eMule peers.

Thus, and given that the behaviour of the currently installed eMule peers may not be changed, we decided to modify the queue score calculation algorithm used by ASR peers in order to favor other ASR peers over eMule peers. We tried several variations of the base algorithm taking into account the goal of maintaining traffic savings. The variation that provided the best performance with the lowest traffic consisted in providing eMule peers within the search radius with a penalty factor divided by their distance (in hops) and those outside the search radius with a tenth of this. This is shown in Eq. (4). ASR peers were provided the original eMule score.

$$eMule\_score = \begin{cases} \frac{penalty \times rating \times file\_priority \times credit \times seconds\_in\_queue}{peer\_distance \times 100}, \\ \quad \text{if } peer\_distance \leqslant SR \\ \frac{penalty \times rating \times file\_priority \times credittimes\ seconds\_in\_queue}{10 \times peer\_distance \times 100}, \\ \quad \text{if } peer\_distance > SR \end{cases} \tag{4}$$

We also found that, since eMule peers will not be as cooperative to ASR peers as other ASR peers, the search radius should be larger in order to encompass more peers. In counting the number of sources for each file part, ASR

Table 8
Partial deployment using revised ASR algorithm.

	10% ASR	90% eMule	50% ASR	50% eMule	90% ASR	10% eMule
Avg. download time (s)	3536	4121	3212	3325	3094	2850
Avg. search radius	10.21		8.47		7.46	
Avg. download time	4063		3269		3070	
Transit traffic (GB)	422		314		178	
Stub traffic (GB)	661		617		539	
Inter-ISP traffic (GB)	209		176		118	
Num. connections (×1000)	651		353		137	
Avg. hops crossed	9.78		8.4		6.47	

was modified to account eMule sources as a fraction of a source, thus resulting in larger search radiuses. The same penalty factor (0.1) used to modify the queue score was used to account eMule sources.

Table 8 summarizes the results obtained with the revised ASR algorithm. The first users to migrate to ASR experience much faster download times, which decrease continuously as more users migrate and ASR is able to use shorter search radii. Simultaneously, the introduction of ASR peers also provides eMule peers with faster download times, even though they are discriminated by ASR peers. The traffic saving properties of ASR are still maintained, although savings are inferior to those of the original ASR protocol, shown in Table 7. Everyone stands to gain from the introduction of the revised ASR peers.

Still, users are only compelled to migrate up to the point when ASR peers become dominant. When 90% of the peers use ASR, those who stuck to eMule will observe much faster downloads. This is explained by ASR peers gathering in small clusters, not asking to download from peers outside their search radii. The remaining eMule peers are able to take advantage of ASR peers which are outside the search radii of other ASR peers, as they ask to download from everyone. As such, in this late stage, further favoring ASR peers provided no results. This situation might prompt some ASR users to return to eMule, ending up with higher download times for everyone and more traffic.

However, should ASR peers completely refuse to upload to eMule peers, these will experience long download times, forcing users to move onto ASR. Table 9 shows what happens in a 90% ASR, 10% eMule population when ASR peers refuse to upload to eMule ones. At the cost of heavier traffic, eMule users experience a long enough download time to prompt them to make the change to ASR.

A migration plan is therefore necessary, to go from today's 100% eMule population to an 100% ASR enabled eMule population. We suggest that newer versions of eMule client software implement the revised ASR algorithm. As people upgrade their software, compelled by the faster downloads, ASR becomes common. When ASR enabled peers become the dominant application type, a second version of ASR enabled eMule clients will start being deployed, which will refuse to upload to legacy eMule peers. As users migrate to these newer versions, those using legacy eMule will start to experience increasing download times and will eventually decide to migrate. This is a feasible migration plan as most eMule clients are distributed as free or open software, and migrating incurs no cost to users.

Table 9
Partial deployment when refusing to service eMule peers.

	90% ASR	10% eMule
Avg. download time (s)	3075.54	6253.65
Avg. search radius	7.63	
Avg. download time	3391	
Transit traffic (GB)	185	
Stub traffic (GB)	534	
Inter-ISP traffic (GB)	121	
Num. connections (×1000)	168	
Avg. hops crossed	6.49	

The impact on those still using legacy eMule would not be as extreme as depicted in Table 9, where all ASR enabled peers refuse to upload to legacy eMule peers. In fact, as the second version of ASR enabled eMule clients is introduced, the large majority still runs the first version. The number of peers refusing to upload to legacy eMule clients would initially be small but gradually increase. Legacy eMule users would be compelled to upgrade before such an extreme scenario was ever reached. An alternative method would be for the second version peers to limit the rate of upload to eMule peers, instead of simply refusing to upload to them. This could slow down the migration process but would probably make it a more acceptable solution.

We validated this migration plan using a multiple file download scenario similar to the one used in Section 5.2. The results supported the conclusions presented in this Section.

We conclude that although beneficial to users if used alone, when used in a mixed environment with plain eMule peers, ASR showed a behaviour which would discourage users from migrating to it. As such we designed and validated a two phase migration plan, which allows ASR to be gradually introduced in a way that users feel compelled to exchange plain eMule clients for ASR clients at any stage of the migration. This migration plan only entails modification to the ASR enabled clients that are introduced at each phase.

7. Internet deployment

We conducted experiments using the Internet and live swarms, in order to validate ASR in a real world deployment. These experiments confirmed the results obtained through simulation.

7.1. Experimental scenario

We implemented ASR on JMule 0.5.8 Beta.[6] ASR was added as an option, so that it could be turned on or off, allowing the same client to be used for comparing eMule against ASR. In order to automate tests, we did not use the graphical interface. Instead, we used JMule as if it was a library, implementing a simple command line interface. This client downloads a single file and terminate as soon as the download reaches the end.

We implemented ASR using the modified algorithm presented in Section 6.2. Namely, we modified the scoring algorithm using Eq. (4) and only accounted the part availability provided by eMule peers as one tenth. The ASR client will still enter as many remote upload queues as it can, but when given a change to download, it will only take it if the remote peer is within the current search radius.

In order to determine the hop distance from other peers, we used a cross-layer approach. A packet filter mechanism was used to intercept TCP SYN packets arriving from other peers. We resorted to jNetPcap 1.3.b4,[7] a java interface to the Libpcap library. When a packet arrives, a callback method is called on the application layer. This method will extract the TTL field from the IP, calculate the hop distance using Eq. (1) and store it in a hashtable. An API enables the ASR implementation to consult the TTL for a given IP.

In order to validate ASR under a real scenario, we decided to use the worst scenario detected during simulation. We tried using a single ASR client to download a file shared by a swarm comprised only of eMule peers. This is an extreme case of the simulation scenario presented in Section 6.2. We chose two popular files: a MP3 file with 7.6 MB and a video file with 700 MB. The MP3 file was, at the time it was first searched for, shared by about 260 peers. Due to its small size, which allowed for a fast download, there were few leechers. The video file was shared by almost 400 peers, 10% of which were leechers.

Our software was run continuously, cyclically downloading both files using both eMule and ASR. A single instance of the software was used, configured to enforce a download limit of 8 Mb/s and an upload limit of 1 Mb/s. These rates are compatible with a residential broadband connection. The computer was connected to the Internet using a 100 Mb/s Ethernet link.

7.2. Result analysis

We focus our analysis of the experiments on the fulfilment of ASR's primary goal: reduce Internet traffic. Download time reduction was also experienced when using ASR. However, due to the nature of the Internet and swarm evolution, ASR gains were not constant. Although it provided faster downloads most of the time, this was not always true. Overall, ASR download times were shorter than those of eMule.

Fig. 13 compares the results obtained for the 700 MB file using eMule and ASR. It traces the Cumulative Distribution Function (CDF) of two measurements: the distance from the peers, the amount of data downloaded from peers at a given distance. Fig. 14 presents the same data for the 7.6 MB file.

We can observe that when eMule is used, the two curves are very approximate matches. Peer distance is not taken into account by eMule when selecting which peers to download from. As such, downloads are uniformly distributed over all the peers.

ASR, on the other hand, strives to download from the closest peers it can. Consequently, we observe that the downloaded data per peer distance line is shifted to the left, clearly indicating the data is downloaded from the closest peers. This enables ASR to reduce the burden of P2P FS on the Internet as fewer links are used.

8. Discussion

Throughout our simulation work we concluded that ASR provides enough advantages that its addition to current eMule implementations would greatly benefit both users and ISPs. In this section we discuss some open issues.

Scalability and robustness. To be used on the Internet, ASR must be at least as scalable and robust as eMule. We believe the scalability of ASR to be limited only by that of eMule. In fact, by having each peer be contacted by fewer other peers, larger swarms should be possible.

In terms of computational effort and additional traffic at each node, ASR adds two tasks to eMule: determine the hop distance to other peers and calculate the search radius. Hop distance measurement may be performed with little overhead as explained in Section 4.2. This overhead may be further reduced by estimating, rather than measuring, some of the distances. For instance, if a new peer has the same 24 bit IP prefix as a known peer, we may assume it has the same hop distance. Finally, the algorithms and data structures required to calculate the search radius present a very small footprint when compared to a full implementation of eMule, which means that ASR will have a negligible impact on its performance.

In terms of robustness, ASR may be affected by wrong hop distance measurements. Wrong estimates of the hop distance to other peers may arise by the use of tunnelling technologies such as Virtual Private Networks where an IP packet may be carried encapsulated over many hops without that fact being reflected in its TTL field. As a consequence, a peer could be more distant than actually measured, which might affect the choice of peers within the search radius. Combining the distance metric with delay may avoid these problems.

Security concerns. By creating small clusters of peers, ASR may increase the effectiveness of certain attacks. A Sybil attack [33], combined with crafting adequate TTL values, may lead a peer to communicate with a single host, which can then provide an effective denial of service attack.

The effectiveness of other methods to subvert the system may also be magnified due to the smaller number of peers used by each client. However, it would be difficult to control a full set of peers as each peer has a different

[6] http://jmule.org/.
[7] http://jnetpcap.com/.

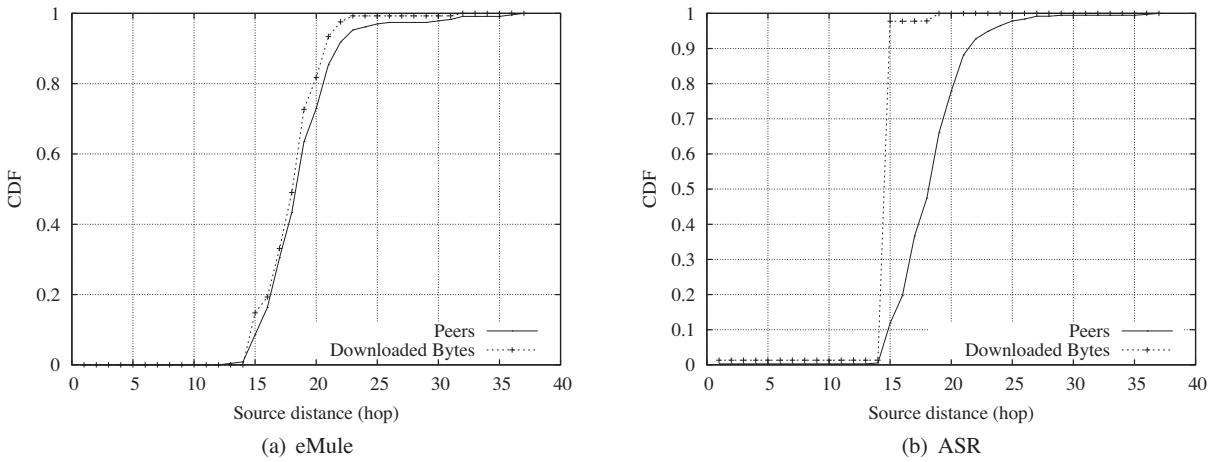


Fig. 13. Cumulative distribution function for number of peers and downloaded data by hop distance for a 700 MB file.

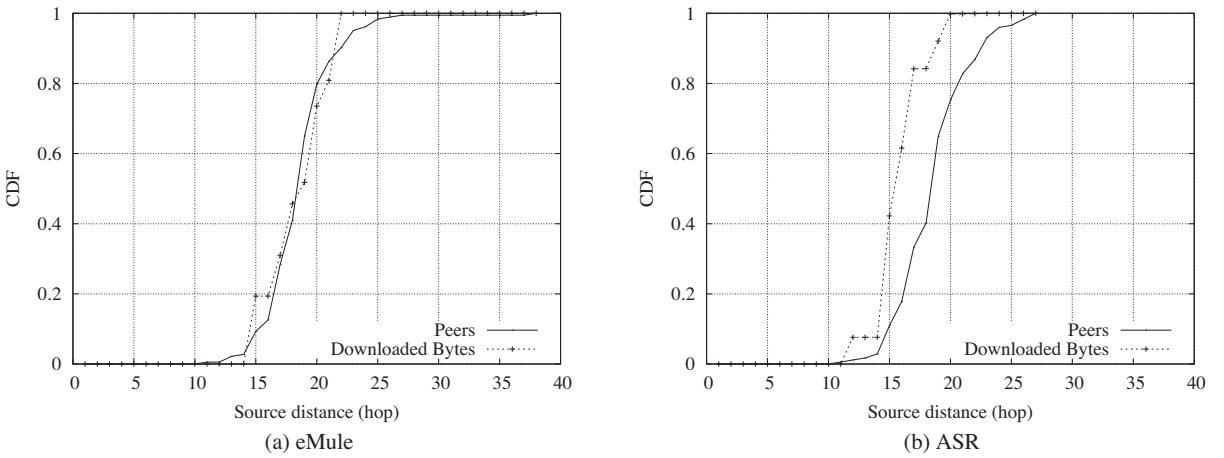


Fig. 14. Cumulative distribution function for number of peers and downloaded data by hop distance for a 7 MB file.

view of the swarm. Except for the case where all the peers fall within the same ISP, the view of the swarm will not be the same for all the peers.

ISP support. We envision that ISPs could have an active role in promoting the use of ASR, by carrying its traffic unrestrained or with fewer restrictions. This would provide a reason for users to migrate, which in turn would result in traffic reductions on the long run. Customers would certainly be more pleased with this than by having the service they pay for restricted by traffic shaping. To limit costs, the benefits could be restricted to links to ISPs with which peering agreements are favorable. ISPs would have to use deep packet inspection to determine which peers announce themselves as ASR enabled.

But possibility and motivation for fraud exists here, with non ASR compliant peers gaining from announcing themselves as ASR peers. To overcome this threat ISPs would have to police the behaviour of peers to detect non compliant ones. This could be achieved by accounting the amount of data exchanged with peers inside and outside the ISP. Some ISPs already perform this accounting for billing purposes.

Closer may not always be better. We have assumed that downloading from close peers is always beneficial. This may not always be the case. A nearby peer may be reachable through a congested link while links to a more distant peer may be idle. However, we believe that in terms of performance, the reciprocal behaviour of P2P FS protocols will avoid congested links. We must only ensure that the search radius of ASR is large enough that there are alternatives to the peers behind the congested links.

ASR does not take into account the nature of each link. An international link is counted as one hop, just like a link between two routers in the same rack. However, international links are usually distant (in hops) from the ISP's customers. International links are usually provided by tier-1 ISPs, and most users connect to tier-2 or higher ISPs. As such, the hop distance between two customers of different ISPs connected by an international link (of a third ISP) is likely to be greater than that between two customers of different ISPs with direct peering agreements.

There is however one scenario with which ASR cannot cope: when intra-ISP links are more expensive than inter-ISP links. This can happen, for instance, when the

last-mile technology is resource constrained (such as 3G cellular networks) or more expensive (when an ISP buys the last mile access from another ISP, like many European ISPs have to buy ADSL links from the incumbent operator) [14]. A possible solution to these problems would be for ISPs to configure routers to decrease the TTL by two or more units when sending packets over expensive links.

Alternatives. A legitimate question to ask is whether ASR is irrelevant or in risk of being obsoleted by other solutions such as traffic caps, price policies, caches, traffic shapers, ALTO or BNS? We believe that despite the introduction of other techniques, ASR has a place of its own.

In the USA, where users have had access to unlimited traffic for years, traffic caps have not been well received by users, and some ISPs have stepped back on their intentions to introduce them. In other countries, such as Portugal, traffic caps, which were the norm up to recently, are being abandoned by most ISPs with the exception of providers of wireless broadband. In places where they exist, a distinction between national and international traffic is sometimes seen, while other ISPs choose to ignore intra-ISP traffic. In these scenarios, ASR provides significant cost savings to users, by limiting the reach of P2P traffic.

Caches are used by ISPs to limit the amount of P2P traffic coming in from outside their networks, or to throttle download rates. However, running a cache may implicate an ISPs in illegal sharing of copyrighted content, subjecting it to legal problems. ASR replaces caches, by forcing peers to prefer local sources, and avoids the throttling imposed by caches. This should make ASR more appealing to users. On the other hand, caches can increase the availability of a file after it is no longer being shared, which cannot be achieved by ASR.

When ASR limits traffic to the inside of an ISP, it helps users avoid traffic shaping at the inter-ISP links. This is advantageous to users and does not increase ISP costs.

ALTO, while still under development, promises to achieve goals that are not covered by ASR, such as allowing ISPs to guide P2P traffic through less congested or less expensive links. However, P2P is still used often to exchange copyrighted content and ALTO requires peers to exchange data with ISP-run servers, sharing information that could be embarrassing or incriminating to users. At a time when copyright holders frequently request ISPs for details about their clients suspected of exchanging copyrighted material, user adoption of ALTO may be undermined or limited to a few P2P applications, especially given that many content producing companies were present in the development of P4P, one of the protocols under consideration. In comparison, ASR does not provide ISPs any more information than the one they already have today.

ASR was designed with different goals than BNS, which explains why they complement each other so well, making BNS less of an alternative and more of an enhancer of ASR (and vice versa). The main differences between ASR and BNS are the results obtained when there are few peers on the same ISP, and the deployment path for both proposals. When compared to BNS, ASR always tries to reduce global Internet traffic by fetching data from close sources, even when it is unable to reduce traffic to transit networks because there are no sources within the same ISP. In this situation, BNS will not change the behavior of the P2P application.

ASR may be used in new P2P applications without requiring ISP intervention. This will allow developers who find ASR interesting to deploy it. BNS on the other hand requires ISP intervention, and its impact will be limited to the ISPs that adopt it. However, BNS will impact all clients of the affected P2P protocol at once.

9. Conclusions and future work

This paper presents a novel approach to the problem of limiting the ever growing P2P FS traffic. We propose a peer selection algorithm that can be used with P2P file sharing applications. This algorithm limits the network distance (in network hops) of the peers that are contacted, causing P2P traffic to cross fewer Internet links. This decreases the number of usable peers, but that fact is compensated by each peer having to serve fewer other peers. ASR is a low overhead solution, easily implementable on file sharing protocols such as eMule. Furthermore, it differs from other solutions by not requiring any ISP intervention.

The simulation analysis performed allowed us to conclude that applying ASR to eMule reduces its impact on Internet traffic, while providing users with shorter download times. ASR proved even more efficient under extremely heavy load conditions, finishing many more downloads than eMule.

We compared ASR to alternatives BNS and traffic shaping. We concluded that ASR and BNS are complementary solutions, which together allow file sharing to be performed faster and more efficiently, benefiting both the user and the ISP. Traffic shaping proved to be the technique which provided the fewer benefits to ISPs while imposing a very high cost on the user download time.

When eMule competes with HTTP for link capacity the latter pays the greatest toll. The use of ASR will not only improve the performance of eMule, but will also improve the performance of other applications, by reducing the impact of P2P traffic on the Internet.

Although ASR is beneficial to users when all peers used it, in a mixed environment with plain eMule peers it showed a behaviour which would discourage users from migrating to it. As such, we designed and validated a two phase migration plan, which allows ASR to be gradually introduced into a plain eMule population in a way that users feel compelled to upgrade plain eMule clients for ASR clients at any stage of the migration. This migration plan only entails modifications to the ASR-enabled clients which are introduced at each phase.

ASR was also evaluated by implementing it on an eMule client software. We used it to download files from live Internet swarms comprised only of eMule peers, an extreme migration scenario, which was found to be the most challenging during simulation. Again, ASR provided significant traffic savings while finishing downloads faster.

In the near future we intend to study the use of ASR with BitTorrent and perform tests on the Internet using the PlanetLab network. This work as already been started.

We also intend to study the combination of search radius as a first peer filter with other metrics, such as latency, which should result in faster download times, without impacting traffic savings. After the set of peers has been reduced by ASR, more expensive algorithms, such as choosing valley-free or minimal AS crossings, become viable.

In its current version, ASR uses a single search radius, calculated using the rarest file part. Should the availability of the rarest part be much lower than that of other parts, these may be downloaded from a greater distance than what is necessary. We intend to study the effect of using several search radii, from just 2 (i.e. one for the 50% most popular file parts and one for the rest) to one for each file part.

References

[1] H. Schulze, K. Mochalski, Internet Study 2008/2009, White Paper (2009). <http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009>.

[2] G. Gentile, The Associated Press, Warner Bros. to distribute movies, TV shows via BitTorrent, USA Today Newspaper (May 2006). <http://www.usatoday.com/tech/products/services/2006-05-09-warner-bros-p2p_x.htm>.

[3] A. Wagner, T. Dübendorfer, L. Hämmerle, B. Plattner, Identifying P2P heavy-hitters from network-flow data, in: Proceedings of the 2nd Annual Workshop on Flow Analysis (FloCon 2005), Pittsburgh, USA, 2005.

[4] Sandvine, Meeting the challenge of today's evasive P2P traffic, White Paper, 2004. <http://www.larryblakeley.com/Articles/p2p/Evasive_P2P_Traffic.pdf>.

[5] PeerApp, Comparing P2P solutions, White Paper March 2007. <http://www.peerapp.com/docs/ComparingP2P.pdf>.

[6] A. Klemm, C. Lindemann, M.K. Vernon, O.P. Waldhorst, Characterizing the query behavior in peer-to-peer file sharing systems, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC 2004), Taormina, Italy, 2004, pp. 55–67.

[7] H. Schulze, K. Mochalski, Internet study 2007, White Paper, 2007. <http://www.ipoque.com/resources/internet-studies/internet-study-2007>.

[8] T. Karagiannis, P. Rodriguez, K. Papagiannaki, Should internet service providers fear peer-assisted content distribution?, in: Proceedings of the ACM SIGCOMM/USENIX Internet Measurement Conference (IMC 2005), Berkeley, USA, 2005.

[9] K. Cho, K. Fukuda, H. Esaki, A. Kato, The impact and implications of the growth in residential user-to-user traffic, in: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2006), Pisa, Italy, 2006.

[10] R.L. Pereira, T. Vazão, R. Rodrigues, Adaptive Search Radius – Lowering Internet P2P File-Sharing Traffic through Self-Restraint, in: Proceedings of the 6th IEEE International Symposium on Network Computing and Applications (IEEE NCA07), Cambridge, USA, 2007.

[11] R.L. Pereira, T. Vazão, On the Impact of P2P file sharing traffic restrictions on user perceived performance, in: Proceedings of the 22nd International Conference on Information Networking (ICOIN2008), Pusan, Korea, 2008.

[12] R.L. Pereira, T. Vazão, On the cohabitation of Adaptive Search Radius enabled peers with regular eMule peers, in: Proceedings of the IEEE International Conference on Communications (ICC 2007), Glasgow, Scotland, 2007.

[13] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, A. Zhangan, Improving traffic locality in BitTorrent via biased neighbor selection, in: Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS 2006), Lisboa, Portugal, 2006.

[14] H. Xie, Y.R. Yang, A. Krishnamurthy, Y. Liu, A. Silberschatz, P4P: provider portal for applications, SIGCOMM Computer Communication Review 38 (4) (2008) 351–362.

[15] C. Griffiths, J. Livingood, R. Woundy, Comcast's ISP Experiences, in: a Recent P4P Technical Trial, IETF draft-livingood-woundy-p4p-experiences-02, 28 October 2008.

[16] V. Aggarwal, O. Akonjang, A. Feldmann, Improving user and ISP experience through ISP-aided P2P locality, in: Proceedings of the IEEE INFOCOM Workshops (INFOCOM 2008), Phoenix, USA, 2008.

[17] J. Seedorf, S. Kiesel, M. Stiemerling, Traffic Localization for P2P-Applications: the ALTO Approach, in: Proceedings of the IEEE Ninth International Conference on Peer-to-Peer Computing (P2P'09), Seattle, USA, 2009.

[18] S.L. Blond, A. Legout, W. Dabbous, Pushing BitTorrent locality to the limit, Technical Report inria-00343822, version 2, INRIA, Sophia Antipolis, France, May 2009.

[19] B. Liu, Y. Cui, Y. Lu, Y. Xue, Locality-Awareness in BitTorrent-like P2P Applications, IEEE Transactions on Multimedia 11 (3) (2009) 361–371.

[20] D.R. Choffnes, F.E. Bustamante, Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems, SIGCOMM Computer Communication Review 38 (4) (2008) 363–374.

[21] M. Piatek, H.V. Madhyastha, J.P. John, A. Krishnamurthy, T. Anderson, Pitfalls for ISP-friendly P2P design, in: Proceedings of the 8th ACM Workshop on Hot Topics in Networks (HotNets'09), New York, USA, 2009.

[22] L. Sheng, H. Wen, Nearby neighbor selection in p2p systems to localize traffic, in: Proceedings of the Fourth International Conference on Internet and Web Applications and Services (ICIW 2009), Venice/Mestre, Italy, 2009.

[23] K.A. Lehmann, M. Kaufmann, Peer-to-Peer Systems and Applications, LNCS, vol. 3485, Springer, 2005. pp. 57–76 (Chapter Random Graphs, Small-Worlds and Scale-Free Networks).

[24] S. Ren, E. Tan, T. Luo, L. Guo, S. Chen, X. Zhang, TopBT: a topology-aware and infrastructure-independent BitTorrent client, in: Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM 2010), San Diego, USA, 2010.

[25] Y. Kulbak, D. Bickson, The eMule Protocol Specification, Technical report, School of Computer Science and Engineering, The Hebrew University of Jerusalem, 17 January 2005.

[26] Hydranode Project, EDonkey2000 protocol. <http://hydranode.com/docs/ed2k/ed2kproto.php>.

[27] T. Miller, Passive OS fingerprinting: details and techniques. <http://www.ouah.org/incosfingerp.htm>.

[28] T. Carstens, Programming with pcap. <http://www.tcpdump.org/pcap.html>.

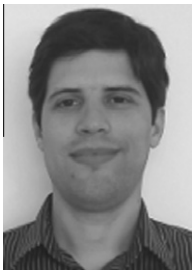
[29] J.H. Cowie, D.M. Nicol, A.T. Ogielski, Modeling the Global Internet, Computing in Science & Engineering 1 (1) (1999) 42–50.

[30] E.W. Zegura, K.L. Calvert, S. Bhattacharjee, How to model an internetwork, in: Proceedings of the Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM'96), San Francisco, USA, 1996.

[31] P.K. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, J. Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), Sagamore, USA, 2003.

[32] V. Cardellini, M. Colajanni, P.S. Yu, Geographic load balancing for scalable distributed Web systems, in: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2000), San Francisco, USA, 2000.

[33] J.R. Douceur, The Sybil Attack, in: Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Revised Papers, LNCS, vol. 2429, Springer, Cambridge, USA, 2002.



Ricardo Lopes Pereira graduated (2000), has a Master (2003) and a Phd (2010) in Computer and Informatics Engineering from the Instituto Superior Técnico (IST) of the Technical University of Lisbon. From 2001 to 2003 he was a Trainee Teaching Assistant at the Department of Informatics and Computer Engineering of IST. In 2003 we became a Teaching Assistant and in 2010 an Assistant Professor. Since 2000, he was been a researcher at INESC-ID where he has worked in areas such as network management, Web an WAP interfaces for network management, active networks, anycast, HTTP load balancing and Peer-to-Peer (P2P). His current interests include routing, load balancing and Peer-to-Peer, with an emphasis on reducing the path stretch of overlay networks and reducing the impact of P2P File Sharing applications on the Internet.

Teresa Vazão graduated in 1985 in Electrical Engineering, at IST. In 1988 she completed her MsC in Electrical Engineering – area of computer networks. In 1999, she finished her PhD in Electrical Engineering – area of computer networks, in IST. She is an Associate Professor at the Department of Electrical and Computer Engineering, at Instituto Superior Técnico, from the Technical University of Lisbon, and is also a researcher at INESC-ID. She has developed research in the field of communications networks, focusing her work on the aspects of the networks architectures. Currently, her research interests are focused in routing protocols in Sensor Networks, Mobile Ad-Hoc Networks and Vehicular Ad-Hoc Networks, as well as Quality of Service Support. She published more than 40 papers in conferences and journals. She has 2 national patents, 2 contributions to the IETF and she is a co-editor of 2 books. She has participated in several Portuguese and European projects, as team member or coordinator of the Inesc-ID team's activity. She is Vice-President of Instituto Superior Técnico for the Taguspark Campus Management since July 2009.

Rodrigo Rodrigues is a tenure-track faculty at the Max Planck Institute for Software Systems (MPI-SWS) where he leads the Dependable Systems Group. Previously, he was an assistant professor at the Instituto Superior Técnico (IST) and a researcher at INESC-ID. He graduated from the Massachusetts Institute of Technology with a doctoral degree in 2005. During his PhD, he was a researcher at MIT's Computer Science and Artificial Intelligence Laboratory, under the supervision of Prof. Barbara Liskov. He received his Master's degree from MIT in 2001, and an undergraduate degree from the IST in 1998. He has won several fellowships and awards, including a best paper award at the 18th ACM Symposium on Operating Systems Principles (SOSP), and a special recognition award from MIT's Department of Electrical Engineering and Computer Science. His primary technical interests are in distributed systems and fault tolerance.