Should Internet Service Providers Fear Peer-Assisted Content Distribution?

Thomas Karagiannis
U.C. Riverside

Pablo Rodriguez
Microsoft Research

Konstantina Papagiannaki
Intel Research Cambridge

Abstract

Recently, peer-to-peer (P2P) networks have emerged as an attractive solution to enable large-scale content distribution without requiring major infrastructure investments. While such P2P solutions appear highly beneficial for content providers and end-users, there seems to be a growing concern among Internet Service Providers (ISPs) that now need to support the distribution cost. In this work, we explore the potential impact of future P2P file delivery mechanisms as seen from three different perspectives: i) the content provider, ii) the ISPs, and iii) individual content consumers. Using a diverse set of measurements including BitTorrent tracker logs and payload packet traces collected at the edge of a 20,000 user access network, we quantify the impact of peer-assisted file delivery on end-user experience and resource consumption. We further compare it with the performance expected from traditional distribution mechanisms based on large server farms and Content Distribution Networks (CDNs).

While existing P2P content distribution solutions may provide significant benefits for content providers and end-consumers in terms of cost and performance, our results demonstrate that they have an adverse impact on ISPs' costs by shifting the associated capacity requirements from the content providers and CDNs to the ISPs themselves. Further, we highlight how simple "locality-aware" P2P delivery solutions can significantly alleviate the induced cost at the ISPs, while providing an overall performance that approximates that of a perfect world-wide caching infrastructure.

1 Introduction

Peer-to-peer (P2P) networks, where commodity personal computers form a cooperative network and share their resources (storage, CPU, bandwidth), have recently emerged as a solution to large scale content distribution without requiring major infrastructure investments. By capitalizing on the bandwidth of end-nodes, content providers that use peer-assisted solutions can benefit from a cost-effective distribution of bandwidth-intensive content to thousands of simultaneous users, both Internet-wide and in private networks.

Peer-assisted solutions are inherently self scalable, in that the bandwidth capacity of the system increases as more nodes arrive: each new node requests service from, but also provides service to, the other nodes. The network can thus spontaneously adapt to the demand by taking advantage of the resources provided by every end-node, thus making it more resilient to "flash crowd" events, which may challenge content distribution networks with hundreds of servers [10]. Overall, the system's capacity grows at the same rate as the demand, creating limitless scalability for a fixed cost.

The best example of such peer-assisted content distribution architectures is BitTorrent, which has been embraced by several content providers (Lindows, Blizzard) to reduce the load from congested servers, minimize the distribution cost, and improve download times of software and patch releases. However, while such peer-assisted architectures can provide significant benefits to end-users and content providers, there seems to be a growing concern among Internet Service Providers (ISPs) regarding the cost of supporting such solutions. Since demand is shifted from data centers to end-nodes, peers become servers for other peers at the network edge. This shift increases the amount of data served by each ISP without a corresponding increase in revenue from the peer-hosted data services provided.

In this paper, we explore the potential impact of future peer-assisted mechanisms as seen from three different perspectives: i) the content provider, ii) the ISPs, and iii) the individual users. In particular, we focus on how peer-assisted solutions affect ISP's traffic as load is shifted from the content provider's data center to peers at the edge of the network. We study how peer-assisted solutions affect ISPs over a range of parameters, and compare them with other solutions such as deploying large server farms or caching-based solutions. To this extent, we use a diverse set of

measurements including BitTorrent tracker logs and payload packet traces collected on the Internet link of an ISP network.

Despite the benefits for content providers and end-consumers, our results demonstrate that peer-assisted content distribution solutions have an adverse impact on ISPs' costs. In particular, we show that current peer-assisted solutions roughly double the total traffic on the ISPs access link as well as their peak load due to the outbound network traffic. The reason is the lack of consideration of peer-assisted algorithms toward optimizing ISP's bandwidth. Such overhead is pushing a number of ISPs toward regulating such traffic, e.g. placing downloading caps. On the other hand, an ISP-friendly protocol that would minimize the ISPs' cost could ease such concerns and prevent providers from blocking or shaping P2P exchanges.

One way of providing an ISP-friendly system is by deploying caches that store the files being requested by the peers. Such a cache system would significantly reduce external network traffic for ISPs. However, caching infrastructures need to be compatible with a wide variety of P2P implementations and require extra hardware and maintenance support. Instead, we consider the possibility of adding small changes to existing peer-assisted distribution algorithms to mimic the performance of a caching solution. In this regard, we highlight how simple "locality-aware" peer-assisted delivery solutions can significantly alleviate the induced cost at the ISPs, while providing an overall performance that approximates that of a world-wide caching infrastructure.

The highlights of our work can be summarized in the following points:

- We provide a detailed study that sheds light on and quantifies the impact of peer-assisted content distribution solutions on ISPs based on real Internet traces.
- We present evidence that establish the potential for locality-aware "peer-assisted" solutions. We estimate and quantify file-availability and user-overlap in time where such solutions are feasible.
- We describe easily deployable architectures for efficient peer-assisted content distribution. For each case, we quantify the benefits and highlight potential savings.

Overall, our work aims at providing ISPs and content providers with a pragmatic, empirical cost-benefit analysis of current and future possible peer-assisted solutions for content distribution.

The remainder of the paper is structured as follows. In Section 2 we make the case for BitTorrent-like systems as P2P-mechanisms that could facilitate large scale content distribution and describe BitTorrent's functionality. In section 3, we demonstrate that current P2P systems and specifically BitTorrent do not exploit locality and present the implications of such a practice on a small ISP. In Section 4 we quantify the impact of P2P content distribution on the capacity requirements of an ISP network. Our findings motivate the need for a locality aware mechanism that directs peers to obtain the requested content from other peers located inside the same network, if they exist. We describe such mechanisms and evaluate their performance in section 5. In section 6, we describe related work. We discuss implications of our findings in section 7 and finally conclude in Section 8.

2 P2P as a mechanism for large scale content distribution

The P2P paradigm appears as an attractive alternative mechanism for large scale distribution of legal content (e.g., *Avalanche* [7]). Traditional content distribution mechanisms typically require vast investments in terms of infrastructure, usually in the form of CDNs and/or large server farms.

However, P2P content distribution is only viable by satisfying the requirements of both the content providers and the end-users.Any newly adopted mechanism should not incur additional overhead on any of the involved parties, i.e. the content provider, the users, and the users' ISPs.

We believe that BitTorrent-like P2P systems present a unique potential in achieving the aforementioned goals. As a P2P protocol, BitTorrent enjoys the benefits of a distributed system that is inherently more robust to events such as flash crowds, shown to be challenging even for CDNs with hundreds of servers [10], as well as cheaper in terms of infrastructure cost on the part of the content provider. Content distribution using BitTorrent has been shown to offer outstanding performance in terms of content delivery rates to the clients [22, 9]. Lastly, BitTorrent features a unique policy across P2P protocols, called the tit-for-tat policy, which is described below and ensures higher content availability.

In detail, the BitTorrent file distribution protocol specifies the functionality of three main entities [28]: a) a *tracker* which acts as a centralized server by coordinating the distribution and receiving information from all connected peers (over HTTP), b) a *torrent* meta-info file with basic description of the specific file (length, hash, name, etc.) and the tracker (IP), and c) the actual peers downloading the file. Peers only serving the file are referred to as "seeds", while downloading peers are called "leechers". Peers connect first to the tracker requesting a list of available peers in the network and then randomly select a subset of them to download from. This "peer" subset is periodically updated based on the contributions of each individual uploader with each peer trying to achieve the maxi-

mum available throughput. BitTorrent is characterized by a "choke/unchoke" mechanism that encourages cooperation among peers (peers upload more to the peers that offer them more data, *tit-for-tat* policy).

The "tit-for-tat" policy is a distinctive advantage of the BitTorrent system which renders it ideal for a P2P content distribution scheme: Peers are forced to always share the content during their downloads while "free-riders" [2] are indirectly banned from the network. Given that unlike today's P2P file-sharing networks, there is no notion of a "community" from the content distribution perspective, the tit-for-tat incentive ensures availability of the content as long as peers are requesting it.

However, BitTorrent users have no incentive of sharing the content once the download is complete. This practice is in contrast to the majority of P2P file-sharing networks where files are often made available even long after the completion of the download. Thus, availability may be compromised in a P2P content distribution scheme if demand is not high enough to ensure a sufficient number of active users. In the latter case, the provider will then have to ensure a larger number of active "seeds" in the network increasing its cost but facilitating availability. Even in this case, the benefits of a peer-assisted solution outweigh the cost compared to traditional content distribution approaches.

In the following sections we study both the effect of peer-assisted content distribution on the resource requirements of ISPs as well as the performance experienced by end-users. We use two different types of real measurement data. First, we study BitTorrent traffic from payload packet traces collected at a 20,000 user access network. These traces provide the view of file availability and potential savings from an edge-network view at small time scales. In addition, we analyze the tracker log for the RedHat v9.0 distribution that spans five months and offers a global perspective to the same problem at larger time scales.

3 P2P Content Distribution: the view from an edge network

In this section, we first look into the overhead of content distribution through BitTorrent as experienced by an edge network. We then quantify the savings that could be gained in the same scenario if locality was exploited. We conclude by examining the implications of locality-aware mechanisms on the user experience.

For this study we use three day-long packet traces collected with the high speed monitoring box described in [17] (Table 1). The monitor is installed on the access-link (full-duplex Gigabit Ethernet link) of a residential university that hosts numerous academic, research, and residential complexes with an approximate population of 20,000 users (a population equivalent of a small ISP). Our monitors capture the TCP/IP header and adequate payload information from all packets crossing the link in both directions to enable the identification of specific applications. P2P traffic accounts for approximately a third of the traffic in each one of the three traces (identified by the methodology described in [11]), while 13%-15% of the total traffic (8%-11% of the total packets) in the link is due to BitTorrent flows. The large fraction of BitTorrent traffic reveals its growing popularity and offers a sufficient sample to study its dynamics.

3.1 Methodology

Studying the dynamics of the BitTorrent network in the traces involves identifying all BitTorrent packets, determining their specific format and reconstructing all interactions across all BitTorrent flows. To identify BitTorrent flows and messages, we have developed a BitTorrent protocol dissector. While the *Ethereal* protocol analyzer [6] has a BitTorrent module, we encountered several problems with missed packets (e.g., TCP/IP packets that contained BitTorrent messages in the payload but were not identified by Ethereal), in the handling of out-of-order and fragmented BitTorrent messages and with multiple messages in the same TCP/IP packet. Furthermore, Ethereal's BitTorrent module cannot dissect tracker HTTP request/responses.

Specifically, we need to identify two types of messages for all BitTorrent flows: the tracker request/responses and the messages between peers. The nominal format for all packets can be found in [28]. The tracker request consists -among other things- of the peer id, the file hash and the local IP of the BitTorrent client (optional). The tracker response is usually a list of available clients for the requested file with statistics regarding the number of seeds, leechers etc. Regarding peer interactions, detecting the following BitTorrent messages is crucial to our analysis:

- *Handshake:* The first BitTorrent message transmitted by the initiator of the connection. It specifies the hash of the file and the *peer id*.
- *Piece:* BitTorrent files are divided into *Pieces*. The size of each piece is usually $262KB - 1MB$ and pieces are further subdivided in *blocks* of typically $16KB$. Blocks constitute the byte-segments that are actually transferred. The *Piece* message contains a data block of a given piece. The first nine bytes of the message specify the piece index, the byte offset within the piece and the size of the block.
- *BitField:* BitField specifies which pieces of the file are available for upload. It is a sequence of bits where set bits correspond to available pieces. It is typically the first message after the handshake.
- *Have:* The *Have* message advertises that the sender has downloaded a piece and the piece is now available for upload.

Table 1: Description of full-payload packet traces

Set	Date	Day	Start	Dur	Direc.	Src.IP	Dst.IP	Packets	Bytes	Aver.Util.	Aver. Flows/5min.
Jan	2004-01-20	Tue	16:50	24.6 h	Bi-dir.	2709 K	2626 K	2308 M	1223 G	110.5 Mbps	596 K
Apr	2004-04-23	Fri	15:40	33.6 h	Bi-dir.	4502 K	5742 K	3402 M	1652 G	109.4 Mbps	570 K
May	2004-05-19	Wed	07:50	28.6 h	Bi-dir.	1246 K	1301 K	3073 M	1706 G	132.5 Mbps	799 K

BitTorrent flows and messages are identified then through the following steps: a) individual packets are classified into flows based on the 5-tuple (source/destination IPs, source/destination TCP ports and protocol), and b) our dissector looks for the BitTorrent handshake message in the first two data packets of a flow (the BitTorrent handshake should be the first *data* packet of a BitTorrent flow but we allow for malformed packets). If the packet contains a BitTorrent handshake then the flow is flagged and all packets are examined by our dissector. All packets of flagged flows are dissected by keeping state of the interactions between the source and destination IPs. Note that the file transferred between the two peers of a BitTorrent flow is only specified at the *Handshake* message. Thus, while identifying a BitTorrent flow without the handshake message is possible, distinguishing the file transferred is not. In our case, this limitation only affects BitTorrent flows that are already active at the beginning of our traces.

Information on all BitTorrent interactions among peers allows us to track user requests, the associated downloads, the amount of time users request the same content at the same time, as well as the potential impact of locality aware peer-assisted content distribution on the utilization of network resources. We will look into each one of these metrics later in this section.

Identifying individual peers per file: Our analysis depends on robust identification of distinct peers in the BitTorrent network. However, identifying individual peers from BitTorrent messages is far from straightforward.

We can identify individual peers using a) tracker request and b) peer handshake messages. Limitations exist in both cases; while some of these restrictions are common, others are specific to each type of message. Thus each individual mechanism may be used to reinforce the accuracy of the other. Consequently, our peer identification relies on both methods. Pitfalls that need to be taken into account are the following :

Network Address Translators (NATs): Peers cannot be identified based solely on the IP address because of NATs, in which multiple peers appear to have the same IP address. To overcome this limitation we couple the IP with the observed *peer id*.

Peer id: The peer id is not unique for the same peer and varies with time and across flows. Typically the peer id comprises two parts: a) the first $6-8$ bytes of the peer id (out of 20 total) reveal the user client and version of the client (e.g, -AZ2202-, Azureus client, version 2.2.0.2), and b) random bytes. The random portion of the peer id changes with time and across flows. Thus, coupling the IP with the peer id, may result in double-counting peers if the random part varies. To avoid this pitfall, we only couple the IP with the *non-random* part of the id (the peer can be safely assumed to use the same BitTorrent client within the time scales of interest - in the rare case where multiple NATed users use the same client and IP address for the *same* file, we will consider them as one, thus underestimating locality which is the main theme of this work). Thus, *a distinct peer is now defined by the IP and non-random portion of the peer id.* Note that the number of bytes describing the non-random portion of the peer id varies with the client (e.g., for Azureus is 8 bytes while for BitComet is 6, etc.) [28].

BitSpirit (BS) client: Even the aforementioned definition of a peer does not guarantee identification of distinct peers due to the peer id assignment algorithm of the *BS* BitTorrent client. *BS* clients employ a function called "Adjust Client Identify", that modifies the non-random portion of the peer id to match the other peer's client in every flow! The BitTorrent client of other peers is known through the tracker responses (tracker responses include a list of peer IPs, ports and peer ids). On the contrary, the random part of the peer id remains constant across flows. This operation of *BS* clients is only specific to peer handshake messages and can be overcome by collapsing all different peer ids from the same IP which present the same random part of peer id into one user. Also, this restriction may be overcome by correlating peers as found by peer handshake messages with those shown by tracker messages for the specific IPs.

Proxies: The source IP of a tracker request does not always correspond to the IP of the peer even when the peer is not behind a NAT. A number of tracker requests are intercepted by *proxy* servers (tracker requests use HTTP) substituting the source IP of the peer with the one of the proxy. We identify such cases by the *proxy_fwd_for* header field (when available) which also reveals the original IP of the TCP packet. To avoid treating proxies as peers, we replace the proxy IP with the IP specified in the *proxy_fwd_for* field.

Random peer ids: A number of clients assigns random peer ids. This case affects mainly tracker requests and reports where the peer id may vary with time. As with the *BS* client restriction, correlating peer ids from handshake and tracker messages disambiguates individual users.

To achieve robust identification of distinct peers, we employ the above methodology in both types of messages (handshakes and tracker requests) separately and compare their outcomes. We found that the agreement between the

two cases was sufficient to discriminate individual users per file. In total, we observed only 3 files out of a total of 360, where the produced list of users per file was not the same across the two methods. In these cases, we selected the list with the least number of users to avoid overestimating potential benefits of locality.

3.2 Hit Ratios

We quantify locality in terms of hit ratios. The hit ratio (analogous to caching hit ratio) refers to content that has already been downloaded and is present locally within our monitored ISP.

We examine the hit ratio along three dimensions:

File hit ratio, where we assume that the complete file is "cached" locally after the first download. Local caching would be the equivalent of a local-aware P2P system, where once a full copy exists within the ISP (either one peer has the full copy, or pieces of the file are spread across the ISP's customers), requests are served locally (assuming always active peers). Thus, the file hit ratio reveals the fraction of multiple downloads of the same content for the ISP in terms of the total number of downloaded files. Let N_i, be the user population for file i within the ISP with n being the total number of files. Then, the file hit ratio is defined as follows:

$$Hit\ Ratio = \frac{\sum_{i=1,n}(N_i - 1)}{\sum_{i=1,n} N_i}$$

Byte hit ratio, where we incorporate the file size in the hit ratio. The byte hit ratio is defined by multiplying each term in the sum of both the numerator and the denominator in the previous fraction by the size of file i (the file size can be inferred by the bitfield and piece messages[1]).

Piece hit ratio, where we examine what fraction of the incoming downloaded pieces for each file existed locally at the time of the download. Local pieces can be inferred by outgoing BitTorrent messages. Thus, there is a "Hit" for a downloaded piece, if the specific piece was advertised earlier in the trace by a local user through the *BitField* or *Have* messages. The total hit ratio is then the fraction of hits divided by the total number of downloaded pieces. Note that while our monitoring point disallows the capture of local interactions (packets transferred within the ISP boundaries not crossing our monitored link), our view of the status of each file is not limited for two reasons: a) the BitField message reflects the current status of the file with every new flow for the same file (BitTorrent protocol is characterized by a large number of open connections which yields a significant number of flows), and b) once a new piece is acquired, the peer advertises the specific piece to all its connected peers with the *Have* message.

[1] FileSize = Number of bits in the *BitField* $\times$ (maximum byte offset in a *Piece* message + block size of the *Piece* message)

Table 2: File, byte and piece hit ratios for the three traces.

	January	April	May
File Hit Ratio	14%	10.4%	18.2%
Byte Hit Ratio	12%	9.6%	13%
Piece Hit Ratio	6%	6%	11.8%

Table 2 presents the three hit ratios for all files across our three traces. The file hit ratio ranges from 10%-18% with the byte hit ratio being slightly lower. Taking into account specific piece and timing information reduces the hit ratio even more (6%-12%). Given the short duration of the traces (one day) and the small size of the monitored ISP the hit ratio is non negligible. Similar observation for one-day file and byte hit ratios have also been presented in [8][24] for the Kazaa network.

Requested files in the network are short-lived in accordance with previous findings in other P2P networks [25]. Only three files existed in both the April and May traces, while the January and April traces had only one file in common. Short-lived files imply that a P2P caching solution would not require large amounts of space and that if P2P nodes were to make their files available for a short period of time after the download is over, nodes could enjoy most of the possible sharing benefits.

3.3 Peer overlap in time

BitTorrent-like P2P systems assume the existence of a large number of *active* end users for a single file; peers participate in the sharing of the content by uploading at the same rate approximately as they download (tit-for-tat). The assumption of simultaneous active users, while valid globally, needs to also be valid within the boundaries of individual ISPs so that locality is beneficial. Note that an "active" peer in BitTorrent implies that the peer is currently downloading/uploading the specific content and not just participating in the network.

We quantify user overlap in time by tracking peer dynamics for all files in the network. We are interested in files that are downloaded/uploaded by at least two local peers throughout our traces, since locality or caching would have no impact on files requested by a single peer. Files with at least two users account for 10.5% (18/172), 8.7% (30/346) and 11.1% (34/306) of the files for our January, April and May traces respectively.

Peer overlap in time ranges from 30%-70% in our traces and is defined as the time during which more than one active users exist for at least one file versus the total time of the trace. Fig. 1 presents time overlap of peers for the April trace. The top line shows the number of "active" files in time; by active, we refer to files for which we observe activity at the specific time instance (download or upload). Note that we only plot files with more than one request over the whole trace. The bottom line shows the number of files with more than one "active" user at each time instance. Ac-

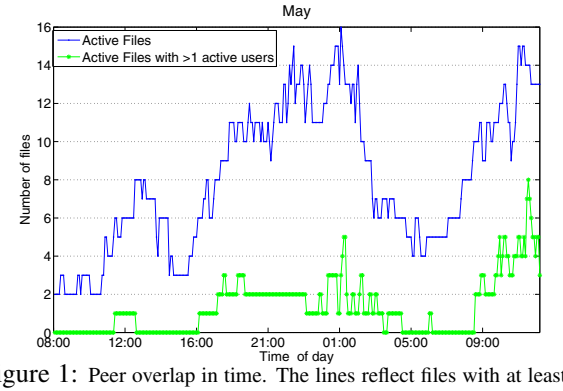


Figure 1: Peer overlap in time. The lines reflect files with at least two users over the whole trace. The top line presents active files in time. The bottom line shows the number of files with at least 2 active users in time. Approximately 60% of the time multiple active users coexist and could cooperate in the distribution of the content.

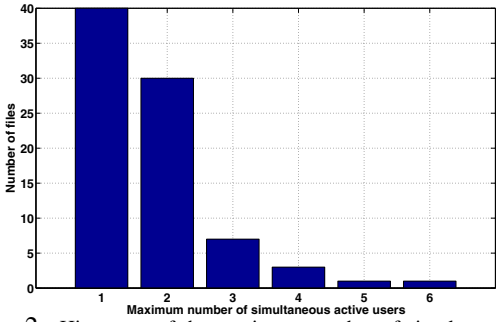


Figure 2: Histogram of the maximum number of simultaneous users per file for all files with at least two users in our traces. User-overlap is present for approximately 50% of the files, while 15% of the files have at least 3 simultaneous users at some point in time.

tive files and users follow the known diurnal patterns while towards the beginning of the second day the number of active users and files increases significantly. Overall, there exists at least one file with at least two active users 25%, 42% and 60% of the time for the January, April and May traces. Accounting for the fact that a number of already active BitTorrent flows at the beginning of our traces may have been missed, the percentages increase by 5%-10% after removing the first 5-10 hours of the traces.

The maximum number of simultaneous active peers per file in our traces is six. Fig. 2 shows the histogram of the maximum number of active users per file for all files requested by at least two peers. Locality could be exploited for 50% of the files where users coincide (we consider time overlap of at least 10 minutes to regard peers as simultaneous). Moreover, we observe at least three simultaneous peers for roughly 15% of the files.

3.4 Potential Savings on ISP bandwidth

Having established the co-existence of active users for the same file, we now quantify the percentage of "unnecessary" downloaded bytes. To estimate potential savings, we assume two scenarios: a) the caching case, where all local pieces are available once downloaded irrespective of the availability of the peer having the piece, and b) the peer-assisted case, where only local pieces in active users are

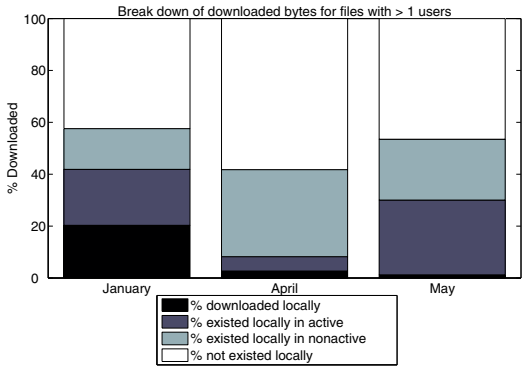


Figure 3: Breakdown of downloaded bytes for files with at least 2 active users. 70%-90% of existing local pieces are downloaded externally while 50%-90% of these pieces exist in active peers.

considered. Local pieces are inferred by the *BitField* and *Have* messages as discussed previously.

70%-90% of existing local pieces and 50%-90% of existing local pieces in active users are downloaded externally! Fig. 3 summarizes our findings by presenting a breakdown of all downloaded bytes for files with $N_i > 1$. Only a minimum portion of bytes is downloaded locally even though more than 20% of the bytes exist in active users (with the exception of April). In an ideal caching scenario, at least 40% of the content exists and could be downloaded locally.

3.5 Performance Implications for the User

Locality-aware peer-assisted content distribution mechanisms may have further benefits in terms of performance for individual peers. First, edge networks may feature much wider bottlenecks than the global Internet. In addition, the number of hops between peers is likely to be smaller and the associated propagation delays shorter, if the traffic stays within the ISP. For instance, a Gigabit Ethernet Local Area Network is likely to offer shorter, higher-throughput paths, to local clients compared to the case where clients are redirected to cross the Internet in order to retrieve the same content. To test this assumption we proceed as follows.

Our packet traces allow us to observe the throughput obtained by each user for each file retrieval at each 5 minute interval. The aforementioned throughput value corresponds to the performance experienced by the user using today's BitTorrent system. In a locality-aware variation of BitTorrent the peer is going to be served by a local peer whenever such a peer is active. Consequently, for these periods of simultaneous activity the peer is going to receive higher throughput than the one measured in our trace. We assume that the throughput offered by local peers is going to be at least as much as the maximum cumulative 5-minute throughput the user achieves throughout the trace, i.e. the peer is capable of matching its maximum upload/download rate. If we call $r(i,t)$ the rate measured in the trace for user i at time interval t, and $R(i)$ the maximum cumulative

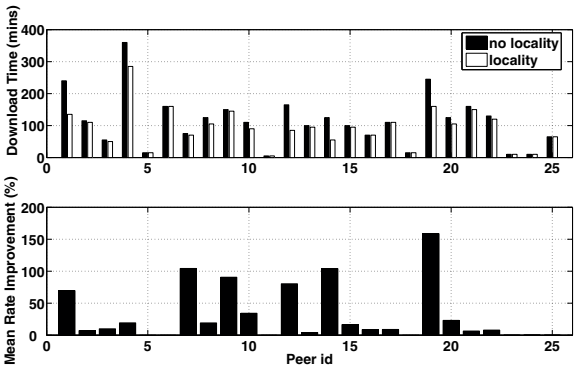


Figure 4: Implications of locality on user performance. The upper plot presents the total download time per user without and with a locality-aware p2p scheme. The lower plot presents the percent improvement in the achieved mean download rate.

rate of user i across the entire trace, then our simulation works as follows: 1) while the cumulative number of bytes downloaded by all peers is smaller than the file size, users download at their measured rates $r(i,t)$, 2) when the cumulative amount of bytes downloaded by the active users exceeds the size of the file, then all active users can download the rest of the file they need at $R(i)$ (a rate which is likely to be lower than the actual rate the user can receive within his own network), 3) when user i finishes the download of the entire file, it leaves the network - the number of active users is reduced by one and the cumulative amount of bytes in the system is reduced by the number of bytes downloaded by the departing user (e.g. the file size); this means that when the last user leaves the network the cumulative amount of bytes in the system goes back to zero. At the end of the simulation we measure the total number of bytes downloaded by each user i, and the amount of time it took for user i to finish the download. We then compute the average throughput each user i could achieve in the locality-aware case.

We ran the simulation experiment described above for the three most popular files in our packet traces. Notice that these files had at least five user requests in a day and at least one user in the Torrent downloads the file in its entirety during the one day period. In Fig. 4, we plot the average latency in the two tested scenarios (locality and no-locality) and the mean rate improvement for each user. Approximately 70% of the peers show increased mean rate in the locality scenario. The improvement ranges from minimal to more than 100% for a particular client, which experiences a mean rate improvement of up to 150%, resulting in a reduction in download time exceeding one hour. 24% of the clients experience more than 50% faster download rate, while 30% of the clients see no improvement with locality since no other user is active at the same time for that particular file and thus the file needs to be downloaded from outside the ISP. Admittedly, the sample provided by our packet traces is very small. However, given that larger networks are more likely to feature a larger pool of simultaneous active users, the associated benefits in download times are likely to be even higher.

Synopsizing our analysis of locality in our packet level traces we observe the following:

- File, byte and piece hit ratios range from 6%-18% in a day, implying that approximately one in ten files is downloaded more than once within the ISP in a day.
- Files are short-lived in the network with only a minimum number of files being requested across months.
- Active users coincide at least 30% of the time and could potentially cooperate in a locality-aware P2P system.
- At least 20% of the downloaded content existed locally in active users, while a minimum of 40% existed locally in both active and nonactive users in two of our traces.
- Peer download rates increase and download completion times per peer decrease for at least 70% of the clients in an idealized peer-assisted scenario.

4 Impact of Peer-Assisted Content Distribution on ISPs: A global perspective

In the previous section we established the benefits of locality aware peer-assisted solutions for a small ISP both in terms of ingress bandwidth as well as user experience. In this section we study the impact of peer-assisted content distribution systems on ISPs and content providers at a larger scale. We assess the overall effects of such a system, by studying the BitTorrent tracker log of the Redhat v9.0 distribution. We analyze and compare traditional client-server distribution approaches with existing P2P and caching techniques, as well as locality-aware peer-assisted mechanisms.

In the remaining of the section, we first provide a description of the tracker log and the analysis methodology, and then present our findings for the various content distribution scenarios.

4.1 Tracker log description and content distribution scenarios

The tracker log spans five months, from April to August 2003, and consists of reports conveying the progress of individual peers in downloading the 1.855GB of the Redhat v9.0 image file. As reported in [9] the user-population exhibits clear flash-crowd behavior.

Each peer sends periodic (typically every 15-30 minutes) updates to the tracker containing the amount of uploaded and downloaded data for the specific interval. Every entry of the log contains the IP address of the sender, a

timestamp and the peer report to the tracker. Peer reports typically contain the following fields: *file hash*, *peer id*, *port*, *downloaded* bytes, *uploaded* bytes, *left* bytes, *event* (started/completed/stopped), *ip* (optional) and *numwant* (preferred length of the requested peer list).

Similar to our packet level analysis, identification of individual peers through the tracker messages is the first essential element in order to explore the impact of the Redhat distribution on the ISPs through the BitTorrent network. However, the procedure of distinguishing peers in the log messages poses further challenges beyond the pitfalls highlighted in section 3. First, the peer id is always random; user clients are not encoded in the peer id as, at the time, there existed one major client, namely the official BitTorrent client (http://www.bittorrent.com/). Second, user-sessions, especially multi-session downloads cannot be reconstructed in a straightforward manner (also discussed in [9]). A session is defined as a sequence of reports with the same IP and peer id which is modified with every paused and resumed session. Thus, calculating downloaded volumes per peer is problematic as the *downloaded-bytes* field refers to session and not overall bytes. Finally, users may change IPs across or within the same session.

To disambiguate individual peers we employ the following methodology:
A user is defined by the *IP and the peer id* in agreement with our practices in section 3. If the local IP exists in the peer report, we replace the IP address of the sender with the local peer IP, only if it does not correspond to private-address space (e.g., 192.168/24, etc.). The source IP address may reflect NATs or proxies and thus the local address is preferred.
To track multi-session downloads originating from the same IP as well as users behind NATs, we correlate the number of *left* bytes across consecutive reports from the same IP and varying peer ids. The left-bytes field signifies the number of bytes left to complete the overall download of the file, decreasing with time for individual peers. Thus, we regard as two separate peers, peer ids for which the number of left bytes is increasing across two consecutive reports with the same IP. On the contrary, if the number of left bytes in the current report is *less or equal* to the previous report, the two peer ids are merged into one (the most recent), potentially underestimating the user population in some cases.
We track peers changing IPs within the same /24 subnet by maintaining a mapping of all peer ids to IPs. All reports with the same peer id originating from the same /24 are treated as one peer (subject to our left-bytes condition above). We observed approximately 50K such reports (approximately 2%). Users are less likely to switch IPs outside /24 net boundaries (at least within the time-scale of minutes); however, note that users switching IPs across different ISPs (e.g., the download resumes at a different location) do not affect our analysis, and in fact they should be treated as distinct peers (downloaded/uploaded bytes affect the current ISP in each case).

After identifying the individual peers inside the network we group IP addresses into ISPs using the ASFinder module from the Coral Reef suite [12]. For the mapping we use BGP tables from RouteViews collected in May and August 2003[2]. Having a global view of where peers are located inside the network we study the impact of the following content distribution scenarios:

- A server, server farm or CDN is responsible for the content distribution (*Scenario 1*). Peers receive the content directly from the server(s) that reside outside the ISP network. Every peer request corresponds to a transfer across the ISP's Internet link.
- A distribution system based on a standard P2P system (*Scenario 2*). Here, content is distributed across peers. Peers are matched based on random selections in the network. However, we take into consideration the timing information existing in the tracker log. Peers are matched only if both of them are active at a specific time interval. Peers become active at the time of their first report in the log or with a *start* event. Similarly, we consider peers inactive after a *stop* event or in the absence of a report for the specific peer within an hour. Timing information and stop/start event reports are vital for two reasons: a) They minimize the probability of double-counting peers (smooth transition to a new peer id), and b) they allow us to reproduce the dynamics of today's popular P2P systems. If matched peers appear both within the boundaries of the same ISP, their download/upload volumes are not considered for the specific time-interval, as they do not incur any cost for the ISP (the transfer is local).
- A distribution system based on a P2P BitTorrent-like system (*Scenarios* 3&4), where peers periodically redefine their peer list based on measurements of the upload throughput offered by other peers in the network. In this case a peer obtains the requested content selecting peers randomly from groups of active users of comparable upload throughput (tit-for-tat policy). The upload bands are based on a per-IP maximum upload throughput as observed throughout the log (scenario 3), or on peer *instantaneous* upload throughputs (scenario 4). Similar to scenario 2, we consider all timing information (active/inactive peers) from the log, and transfers within the boundaries of an ISP are not taken into consideration.
- A peer-*assisted* content distribution system exploiting locality within the ISP's boundaries (*Scenario 5*). In this scenario, a peer requesting content will be redirected to any active peer (if available) within the same

[2]The use of different routing tables does not impact our findings.

(a) Scenario 1 (b) Scenarios 2,3,4 (c) Scenario 5 (d) Scenario 6

Figure 5: Graphical representation of the examined scenarios. a) Each new download incurs additional cost for the ISP while the content provider uploads as many copies as individual users. b) The content provider uploads less copies shifting the distribution cost to the ISPs through P2P. c) An idealized peer-assisted solution ensures that one copy is downloaded per ISP while users cooperate internally. d) A caching solution where a copy is downloaded per ISP and all users are served from the local cache.

ISP (we discuss such mechanisms in Section 5). Assuming that multiple active users within an ISP have the ability to cooperate, we nominate a "leader" peer that first acquires the content and serves the rest of local users. The leader is the peer with the most available content (inferred by the left-bytes field) at every interval. Thus, when a leader is active within an ISP all other local peers are served by the leader. Note that the leader is more of a conceptual entity for byte-tracking purposes in the log, than an actual implementation of a peer-assisted solution. In reality, such a model assumes that peers have different, overlapping or not pieces of the file and cooperate by serving each other; an assumption which should not be far from reality since downloaded pieces are chosen randomly.

- A distributed caching architecture (*Scenario 6*) resembling a perfect content distribution network with caches at the access link(s) of each ISP. In this scenario, we assume infinite-space caches at the edge of the network, resulting in only one downloaded-copy of the content per ISP. Clients requesting the content are served locally by the cache. Thus, only the first download for each ISP incurs cost and will be considered. Caching would be the equivalent of a perfect peer-assisted distribution scenario, where local peers are always active and can satisfy all requests.

Fig. 5 graphically depicts the various scenarios. The content provider shifts the cost of distribution to end-nodes of the network with P2P (Fig. 5(a)&(b)), while locality mechanisms (peer-assisted Fig. 5(c) or caching Fig. 5(d)) provide savings for ingress ISP traffic.

4.2 Evaluation of content distribution scenarios

We now evaluate the cost and benefits for the content provider and the ISPs for all scenarios described in the previous section. First, we provide the evaluation metrics and a brief overview of our findings, and then present extensive results for each case.

4.2.1 Metrics and Overview

We study the benefits and costs in terms of traffic volumes for the content provider and the ISPs. All scenarios are evaluated under the same set of metrics.

To quantify the ISP cost, we measure the total ingress/egress bandwidth consumed, as well as the 95^{th} percentile of traffic in hourly intervals as observed on their access links. Such metrics are of significant interest to ISPs since they typically translate to monetary costs. Scheme performance is summarized across ISPs using the average value of the obtained distributions.

Content provider cost is measured in terms of total traffic served which corresponds to the bandwidth requirements that the provider should meet in each case. As with the ISP cost, we also measure the 95^{th} percentile of the total traffic served per hourly interval. Our findings can be summarized in the following points:

- Peer-assisted content distribution reduces ISP downlink bandwidth by a factor of two.
- A P2P locality case only requires 1.5 times the peak capacity required by a perfect caching algorithm.
- ISPs are required to upload just over only one copy of the content satisfying at the same time a large number of local peers with a local-aware solution.
- ISP savings in the peer-assisted scenario increase roughly linearly to the logarithm of active users.
- The overhead of a peer-assisted approach in terms of peak load as defined by the 95^{th} percentile, is minimal and in some cases even less than the respective overhead of a caching solution.

In the remainder of this section we expand on the above findings.

4.2.2 Impact on Downloaded Traffic

In this section, we consider the impact that peer-assisted content distribution has on the ISP's downstream traffic. A peer-assisted distribution scheme can lead to downlink bandwidth savings only if there are multiple peers concurrently downloading the same file within the ISP. If no active peers exist for the same file or the matching algorithm does not have the ability to account for them, then the benefits of peer-assisted distribution, in terms of downstream bandwidth, can be significantly reduced.

Table 3 presents the average value of the total and the 95^{th} percentile of the data downloaded by each ISP for our 6 content distribution scenarios and for both May and

August BGP tables (AS1 and AS2 respectively). P2P algorithms that match nodes at random (scenario 2) provide very little benefit in terms of ISP bandwidth savings compared to client/server distribution. In fact, a P2P algorithm with random peer matching provides less than 2% bandwidth savings over the case where the same file is distributed once for each client.

However, current P2P systems such as BitTorrent do not rely on a completely random matching of nodes. As discussed in previous sections, BitTorrent features an algorithm that leads to peers clustering according to their upload capacities. Such a P2P algorithm may result in locality, if nodes within an ISP were to have similar download/upload speeds.

Table 3 shows that BitTorrent-like systems improve the locality as more local clients are matched, however, the extra benefits compared to a completely random selection are almost negligible. A potential reason is that the user population for each throughput band is quite large, ergo random peer selection within the band is more likely to lead to a peer outside the ISP.

On the contrary, peer-assisted locality solutions offer high potential benefits. Our results show that *a peer-assisted locality-aware scheme can reduce the ISP's ingress link utilization by a factor of two*.

Finally, the perfect caching solution provides the best possible benefits for the ISP since only one copy of the file is downloaded regardless of the number of internal requests. Compared to the caching infrastructure, the peer-assisted locality-aware solution results in several times the optimal bandwidth requirements, since peers are not always active resulting in multiple downloads per ISP. Nevertheless, one should note that in absolute terms, the amount of traffic generated by such a peer-assisted scheme is only a small fraction of what a client/server solution would produce.

In terms of the 95^{th} percentile, Table 3 demonstrates that locality-aware solutions perform much closer to a caching infrastructure. In most cases, *a P2P solution requires only approximately* 1.5 *times the amount of peak capacity required by a caching solution*. P2P systems are most helpful when demand is high since a large number of simultaneous users provides increasing opportunities for cooperation. However, when user populations is low, P2P systems like BitTorrent almost revert to a client/server model. Thus, P2P systems even though they do not operate at the optimal when decreasing the total amount of traffic, they appear most beneficial when they are needed the most (e.g. during peak loads).

4.2.3 Impact on Uploaded Traffic

Since peers become servers in the P2P paradigm, ISPs are bound to observe increasing amounts of egress traffic, which may considerably impact their bandwidth cost.

Table 3: Downloaded data (in MB) by each ISP. Percentages show savings compared to the client/server model.

	AS1 (Avrg)	AS2 (Avrg)	AS1 (95^{th})	AS2 (95^{th})
Sc.1	14137	15313	804	862
Sc.2	13954 (1.3%)	15110 (1.3%)	794 (1.3%)	854 (1%)
Sc.3	13784 (2.5%)	14920 (2.6%)	786 (2.2%)	843 (2.3%)
Sc.4	13872 (1.9%)	15023 (1.9%)	791 (1.7%)	852 (1.1%)
Sc.5	6710 (52.5%)	7243 (52.7%)	625 (22.3%)	688 (20.2%)
Sc.6	1191 (91.6%)	1268 (91.7%)	459 (42.9%)	490 (43.2%)

Table 4: Average uploaded data (in MB) by each ISP.

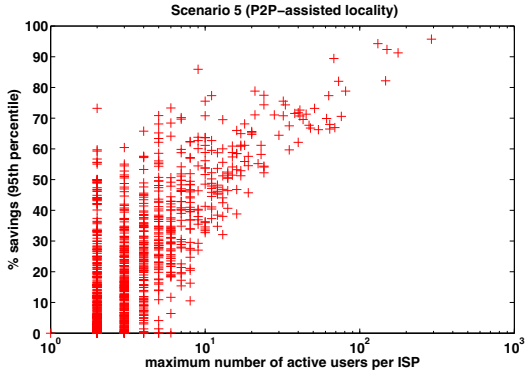
	AS1 (avg)	AS2 (avg)	AS1 (95^{th})	AS2 (95^{th})
Client/Server	-	-	-	-
P2P Random	17239	18188	750	789
P2P BitT	17551	18538	759	808
P2P Locality	2827	2971	238	248
savings	84%	84%	68%	69%
Caching	-	-	-	-

This imposed cost is evident in table 4, where simple P2P content distribution results in high upstream bandwidth requirements compared to the traditional client server model (or the caching infrastructure) where local users do not serve content.

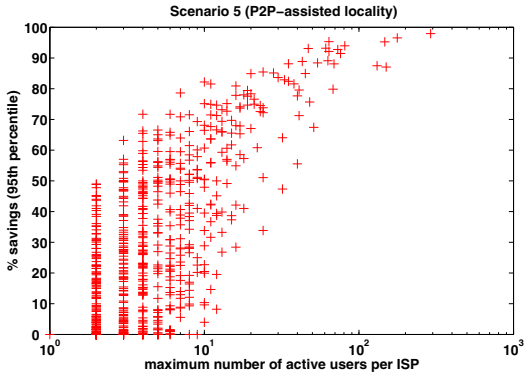
On the contrary, locality keeps most of the traffic within the ISP's boundaries while the amount of traffic uploaded externally is reduced by more than a factor of 6. *In fact most ISPs only need to upload slightly over one copy of the file in order to satisfy a large number of internal users*. Examination of the 95^{th} percentile offers similar observations when comparing current P2P with locality aware systems. However, cross-examination of the peak capacity across tables 3 and 4 shows that the peak upload capacity required is much smaller than the download capacity.

4.2.4 Impact vs. ISP size

We now examine how peer-assisted content distribution affects ISPs depending on their size. Fig. 6 shows the savings in terms of total traffic and the 95^{th} percentile offered by a perfect peer-assisted locality-aware solution compared to a client-server solution depending on the maximum number of active users inside an ISP (which is a rough indication of the ISP's size). Fig. 6(a) shows that savings from the peer-assisted locality-aware solution increase almost linearly with the logarithm of the number of active users. While small ISPs do not experience high benefits as expected, medium and large-size ISPs greatly benefit from the peer-assisted locality-aware system. *In fact, the benefits for ISPs with more than thirty maximum active users are higher than* 60%. In terms of the 95^{th} percentile, the benefits are even higher for the same population size; for a system with a maximum number of active users equal to thirty, the benefits are approximately 80% (Fig. 6(b)).

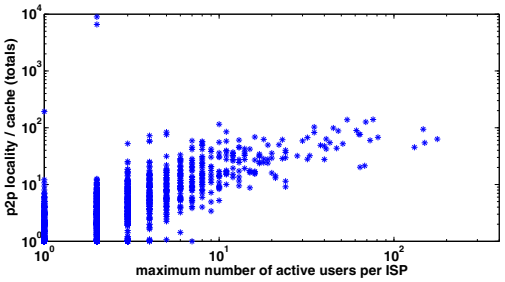


(a) Savings vs. maximum number of active users

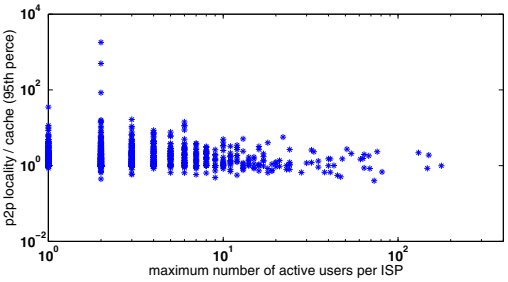


(b) 95th percentile of savings vs. maximum number of active users

Figure 6: Savings of a peer-assisted distribution versus the traditional client/server model in terms of total traffic and the 95^{th} percentile with respect to the ISP size (maximum number of active users). ISPs with more than thirty active users experience savings of 60% and 80% for total and peak traffic respectively.



(a) Overhead of a locality-aware algorithm over a perfect caching system as a function of the ISP's size



(b) Overhead of a locality-aware algorithm over a perfect caching system as a function of the ISP's size (95th percentile)

Figure 7: Comparison of the peer-assisted distribution versus a perfect caching infrastructure. While in terms of total traffic the overhead appears high, the two scenarios appear equivalent in terms of peak traffic. The overhead in terms of total traffic flattens out with increasing ISP size.

4.2.5 P2P Locality-Aware vs Proxy Caches

We now examine how optimal the peer-assisted solution appears compared to per-ISP caching.

Figure 7 presents the ratio of bytes downloaded by a peer-assisted solution versus a caching solution. The overhead of a peer-assisted solution can be quite high compared to the caching solution for small ISPs, since the request rate is low. As the ISP size increases, the overhead of the peer-assisted solution also increases and large ISPs need to download a larger number of copies compared to a caching solution. However, the overhead appears to flatten out at large ISP sizes since large number of active users ensures cooperation.

On the contrary the overhead of a a locality-aware solution is small compared to caching in terms of the 95^{th} percentile. Indeed, *the 95^{th} percentile of the peer-assisted solution is equal to or even lower from the one of the caching scenario*. The lower 95^{th} percentile in the peer-assisted case results from the fact that most peak rates observed in the peer-assisted solution are lower than the peak rate of the caching scheme, thus pushing the percentile closer to the mean rate.

Table 5: Total egress server capacity

	AS1 (avg)	AS2 (avg)	AS1 (95^{th})	AS2 (95^{th})
Client-Server	59.8 TB	59.8 TB	17 TB	17 TB
P2P Locality	28.4 TB	28.3 TB	8.1 TB	8.1 TB
savings	52.5%	52.7%	52.3%	52.3%
Caching	5 TB	5 TB	1.6 TB	1.6 TB
savings	91.6%	91.7%	91%	91%

4.2.6 Impact of Locality on the Content Provider

Finally, we consider the reduction in bandwidth enjoyed by the content provider when a peer-assisted locality-aware solution is used. To estimate the amount of data served by the content provider in the peer-assisted case, we assume that requests which are not satisfied within an ISP need to be satisfied by the content provider. This is an upper bound of the amount of bandwidth required from the content provider since a fraction of the requests would also be satisfied by other ISPs in a P2P system.

Table 5 demonstrates that *locality results in less than half the resource requirements when compared to the client-server model both in terms of total egress traffic and 95^{th} percentile*. Further, a caching solution reduces the total egress capacity by one order of magnitude.

Table 6: Downloaded data (in MB) by each ISP for different locality algorithms.

	/24	/16	DOMAIN	Hierarchical	Proxy Tracker
P2P Locality (Avrg)	13964 (1.2%)	11643 (17.7%)	10864 (23.1%)	10227 (27.5%)	6710 (52.5%)
P2P Locality (95^{th})	779 (3.1%)	698 (13.2%)	709 (11.8%)	689 (14.3%)	625 (22.3%)

Table 7: Uploaded data (in MB) for each ISP.

	/24	/16	DOMAIN	Proxy Tracker
P2P Loc. (Avrg)	5415	4656	4735	2827
P2P Loc. (95^{th})	317	286	289	238

5 Locality Algorithms and their Performance

In this section we describe two simple potential implementations of a peer-assisted content distribution mechanism and compare their performance. Locality may be implemented either by ISPs or be imposed by the content provider. ISPs may implement a locality-aware BitTorrent-like system by deploying *proxy-trackers* at the edges of their network. Proxy-trackers will then intercept requests of local peers and redirect them to any existing active users within the boundaries of the ISP. This practice corresponds to our peer-assisted locality scenario which we extensively studied throughout the previous section. However, such infrastructure-based solutions are not always possible to deploy and maintain.

Alternative locality-based algorithms may be supported by the content provider without relying on extra infrastructure being deployed. Such locality solutions could be implemented with small modifications to BitTorrent trackers and could consider algorithms based on simple prefix rules, domain names, or more sophisticated hierarchical prefix matching rules, network-aware clustering [13] or routing table lookups (the two latter cases would lead to performance very similar to the proxy-tracker scenario).

To evaluate the performance of such algorithms, we assume a certain division of clients among ISPs based on AS information from our May BGP table (results are similar for August). Then, we compare the performance of different locality algorithms versus a perfect P2P locality-aware system that would utilize a proxy-tracker at the edge of each ISP. Note however that the resulting matching of peers in the peer-assisted scheme will be inefficient when the imposed locality algorithm does not match AS boundaries (for example a locality algorithm based on a /16 prefix matching).

Tables 7 and 6 show the amount of data downloaded and uploaded by each ISP depending on the locality algorithm used. The percentage results in brackets are the savings with respect to a client/server solution. Table 6 shows that locality solutions that try to match clients within the same DNS domain are not as efficient as proxy-tracker solutions, however, they provide roughly 50% of the overall benefits. The limitations of such a solution arise from the fact that different clients in the same domain may span multiple ASes, thus, generating large amount of cross-over traffic among ISPs.

Static prefix-based solutions (e.g. /24, /16) that match users within a certain prefix perform overall slightly worse than a DNS based solution. Small prefixes (e.g. /24) result in creating small groups with a limited population of active users, while on the contrary, very large prefixes result in users being too spread across multiple ISPs. Our results indicate that the best prefix-based grouping is /13, which performs slightly better than the domain-matching case.

Finally, we also examine the performance of a hierarchical-prefix matching solution where users are first matched within a /24 prefix, then, /16, /14, and finally /13. The advantages of such a hierarchical solution would be that a) peer-matching can dynamically accommodate ISPs of different sizes, and b) nearby peers within an ISP are matched first. The hierarchical-matching algorithm provides the best performance after a proxy-tracker matching scheme, although, it still fails to match a large number of local peers. This may be due to the fact that ASes typically own non-consecutive ranges of IP addresses, a practice that decreases the effectiveness of any prefix-matching algorithm.

6 Related Work

Previous work on BitTorrent has focused on measurements [4, 9], theoretical analysis [22], and improvements [27]. Izal et al. analyze the log of a BitTorrent tracker showing the flash-crowd effect of a single file, download speeds, and the amount of time that peers stay after they have completed the download [9]. Pouwelse et al. present an extensive analysis of BitTorrent showing availability, peer uptimes, and providing a better understanding of peer inter-arrival times [21].

Apart from BitTorrent, several measurement studies have addressed the issues of availability [2, 3, 8], integrity [29], flash-crowds [4][14], and download performance [1][26][25][3] in other P2P systems. Saroiu et al. use SProbe (sprobe.cs.washington.edu) to measure the bandwidth of Gnutella peers [25]. Liang et al. provide an extensive study of the performance of the KaZaA network [15]. An analysis of Gnutella traces in terms of resource demand, popularity of particular search terms, overlay topology, and node latency was presented by Nogueira et al. [18]. Gnutella data, was also examined by Ripeanu and Foster [23], focusing on node connectivity, overlay topology, and protocol message overhead. A trace analysis

of the network traffic from the perspective of traffic characterization and bandwidth provisioning was presented by Sen and Wang [26]. Markatos [16] conducted a study of Gnutella protocol traffic aimed at caching query/response traffic to reduce network load. Leibowitz et al [14] examined Kazaa traffic to determine proportion of total network traffic by file popularity.

To the best of our knowledge, this paper presents the first rigorous study of the impact that peer-assisted content distribution solutions have on ISPs from both a local and a global perspective. The fact that users in peer-assisted solutions only form a sharing community only while downloading the same file, significantly differentiates them from other existing file-sharing applications and has important implications in the potential benefit of locality-based solutions. For instance, [24] provides an extensive analysis of content delivery systems, including CDN caching, KaZaa, and Gnutella. However, their results do not carry over well to peer-assisted solutions such as BitTorrent where cooperation only happens if clients are active and sharing the same file.

Regarding the locality analysis, previous studies have proposed new ways of clustering peers (e.g. [5][19][20]) and studied the potential benefits of locality in P2P file-sharing systems such as KaZaa and Gnutella [8][24]. In this work, we quantify the potential benefits of peer-assisted locality-aware solutions in a real setting and study the benefits of very simple locality algorithms that require minor changes to existing solutions. Moreover, we study the potential benefit that locality-based solutions have for the content provider and the clients in terms of bandwidth reduction and decreased download times respectively.

7 Discussion

Our analysis thus far has presented a detailed description of the cost and benefits of locality-aware peer-assisted content distribution. However, we would like to stress here a number of implications based on our findings.

Global vs. local benefits: Our analysis presents two different perspectives of the potential long-term benefits of peer-assisted content distribution. While in the global case total savings appear significant, locality appears to only offer minimal savings in our monitored network. This discrepancy is due to various factors, some inherent to each case, others reflecting our measurement data. First, since savings depend on the ISP size (section 4), benefits are limited for our monitored network (smaller than typical ISPs with a limited number of simultaneous active users). Further, while the tracker log covers a period of six months (including the initial flash crowd effect), the duration of all our packet traces is roughly a day thus hiding potential benefits. Finally, locality analysis for our monitored ISP reflects a large number of files (unpopular files or files past the flash crowd effect reduce the observed hit ratios) versus one tracker in the Redhat log data.

Peer-assisted vs. existing content distribution solutions: Our analysis does not in any way suggest that peer-assisted solutions should replace current content distribution practices such as CDNs. Issues such as file availability with small user population, limited end-to-end connectivity with NATs, security, reliability and service guarantees need to be robustly resolved before such solutions can be deployed in a commercial setting. In the meantime, our findings suggest that peer-assisted schemes will definitely prove beneficial when complementing current practices.

Impact of peer-assisted content distribution on internal ISP traffic: While locality-aware peer-assisted solutions appear attractive for ISPs, we would like to stress here that decreasing egress traffic may create the need for traffic re-engineering within ISP boundaries to account for the additional internal upload traffic. Increased upload traffic may prove especially important for those ISPs that depend on other carriers to deliver last-mile traffic (e.g., European Tier-2 ISPs). In fact, for such ISPs, utilizing their transit egress capacity might result in a more cost-effective solution rather than re-routing traffic internally. Caching, on the contrary, fits naturally with the traditional asymmetric architecture of today's ISPs, where the downstream channel is more heavily provisioned relative to the upstream, driven by the assumption that customers download more than what they upload.

8 Conclusions

Based on payload packet traces as well as tracker-based logs, we have studied the impact that local-aware peer-assisted content distribution solutions can have on ISPs, content providers, and end-users. In particular, we have identified that current P2P solutions are very ISP-unfriendly, generating large amount of unnecessary traffic both downstream as well as upstream.

We studied locality in the context of BitTorrent. Our traces indicate that BitTorrent is locality-unaware, severely increasing ISPs' bandwidth requirements. In particular, up to 70-90% of *existing local* content was found to be downloaded from external peers.

In BitTorrent, users typically only share content while their download is active. Our results show that such a feature does not significantly impact the benefits of a BitTorrent-like solution. In fact, we found that users requesting the same file within an ISP overlap 30%-70% of the time and could, therefore, efficiently help each other if a locality algorithm were in place. Furthermore, by having users stay longer after the download is complete and share their content, the potential benefit of a locality algorithm would be an extra 20%-40% in terms of reduced downloaded bytes by the ISP.

Peer-assisted content distribution incurs significant upstream capacity costs for the transit links (roughly doubling the bandwidth requirements). However, simple locality based mechanisms can rectify this effect, approximating the performance of a perfect caching architecture. Overall, locality-aware peer-assisted algorithms decrease the bandwidth of the content provider's egress link by more than a factor of two. Strategies such as those used by BitTorrent trying to match users with similar capacities provide little locality benefits.

The benefits of a peer-assisted locality solution increase with the logarithm of the number of active users. Our findings show that as soon as there are more than 30 active users within an ISP, a peer-assisted locality solution provides more than 60% savings in terms of ISP's ingress traffic compared to a client/server distribution.

On the contrary, a peer-assisted locality-aware solution generates five times more traffic on average through the ISP's link than a perfect caching solution. However, in absolute terms this represents only a small fraction of the traffic generated by a client/server solution.

The benefits of a peer-assisted solution are always much more pronounced in terms of 95^{th} percentile, thus, absorbing peak loads and reducing the monetary impact on ISPs and content providers. Simple locality-aware mechanisms based on domain-name grouping, or prefix grouping provide roughly 50% of the potential benefits.

Our study shows that while current peer-assisted content distribution solutions are ISP-unfriendly, this is not a fundamental limitation and that minor modifications can indeed significantly reduce the costs of all parties involved in the content distribution process. Such simple modifications to peer-assisted protocols can provide a cost-effective solution that can be exploited by content providers to scale and accelerate the delivery of content to millions of users without pushing ISPs towards regulating or blocking such traffic.

Acknowledgments

The authors are thankful to Ernst W. Biersack for providing the RedHat tracker log and to Andrew Moore for facilitating this study.

References

[1] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[2] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In *International Workshop on Peer to Peer Systems*, 2003.

[3] J. Chu, K. Labonte, , and B. Levine. Availability and locality measurements of peer-to-peer file systems. In *ITCom: Scalability and Traffic Control in IP Networks*, 2002.

[4] B. Cohen. Incentives build robustness in Bittorrent. . In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[5] M. Costa, M. Castro, A. Rowstron, , and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *ICDCS*, 2004.

[6] Ethereal. http://www.ethereal.com/.

[7] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. In *IEEE/INFOCOM*, 2005.

[8] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *SOSP*, 2003.

[9] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM*, 2004.

[10] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *WWW*, 2002.

[11] T. Karagiannis, A.Broido, M. Faloutsos, and kc claffy. Transport layer identification of P2P traffic. In *ACM Sigcomm IMC*, 2004.

[12] K. Keys, D. Moore, R. Koga, E. Lagache, M. Tesch, and k. claffy. The architecture of the CoralReef: Internet Traffic monitoring software suite. In *PAM*, 2001.

[13] B. Krishnamurthy and J. Wang. On network-aware clustering of web clients. In *SIGCOMM*, 2000.

[14] N. Leibowitz, M. Ripeanu, and A. Wierzbicki. Deconstructing the kazaa network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, 2003.

[15] J. Liang, R. Kumar, and K. W. Ross. The KaZaA Overlay: A Measurement Study. In *Computer Networks (Special Issue on Overlays)*, to appear.

[16] E. Markatos. Tracing a large-scale peer to peer system: an hour in the life of gnutella. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[17] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt. Architecture of a Network Monitor. In *PAM*, 2003.

[18] D. Nogueira, L. Rocha, J. Santos, P. Araujo, V. Almeida, and W. Meira. A methodology for workload characterization of file-sharing peer-to-peer networks. In *IEEE Workshop of Workload Characterization*, 2004.

[19] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. In *ACM Sigcomm*, 2001.

[20] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for scalable distributed location. In *IPTPS*, 2003.

[21] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P File-sharing System: Measurements and Analysis. In *IPTPS*, 2005.

[22] D. Qiu and R. Srikant. Modeling and performance analysis of bit torrent-like peer-to-peer networks. In *ACM Sigcomm*, 2004.

[23] M. Ripeanu and I. Foster. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In *IPTPS*, 2002.

[24] S. Saroiu, K. Gummadi, R. Dunn, S. Gribble, and H. Levy. An Analysis of Internet Content Delivery Systems . In *OSDI*, 2002.

[25] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *MMCN*, 2002.

[26] S. Sen and J. Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. In *ACM SIGCOMM IMW*, 2002.

[27] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. In *Infocom*, 2004.

[28] Bittorrent Protocol Specification v1.0. http://wiki.theory.org/BitTorrentSpecification.

[29] H. Weatherspoon and J. Kubiatowicz. Naming and Integrity: Self-Verifying Data in Peer-to-Peer Systems. In *FuDiCo*, 2002.