Pushing BitTorrent Locality to the Limit

Stevens Le Blond, Arnaud Legout, Walid Dabbous

Pushing BitTorrent Locality to the Limit

Stevens Le Blond, Arnaud Legout and Walid Dabbous
I.N.R.I.A. Sophia-Antipolis Méditerranée
France

Contact: {stevens.le_blond, arnaud.legout, walid.dabbous}@inria.fr

ABSTRACT

Peer-to-peer locality has recently raised a lot of interest in the community. Indeed, whereas peer-to-peer content distribution enables financial saving for the content providers who do not have to maintain a dedicated infrastructure, it dramatically increases the traffic on inter-ISP links.

To solve this issue, the idea to keep a fraction of the peer-to-peer traffic local to each ISP was introduced a few years ago. Since then, peer-to-peer solutions exploiting locality have been introduced. However, several fundamental issues on locality still need to be explored. For instance, how far can we push locality for a peer-to-peer distribution without impacting its robustness?

In this paper, we perform extensive experiments on a controlled environment with up to 10 000 peers to evaluate the impact of locality on inter-ISP links traffic and peers download completion time. In particular, we show that high locality values enable up to two orders of magnitude saving on inter-ISP links without any significant impact on peers download completion time.

1. INTRODUCTION

Content distribution is today at the core of the services provided by the Internet. However, distributing content to a large audience is costly with a classical client-server or CDN solution. This is the reason why content providers start to move to peer-to-peer content distribution that allows to significantly reduce their cost without penalizing the experience of users. One striking example is iPlayer, a peer-to-peer service for video-on-demand that distributes recent BBC programs.

However, whereas the current peer-to-peer content distribution solutions like BitTorrent are very efficient, they generate a huge amount of traffic on inter-ISP links. Indeed, in BitTorrent, each peer that downloads a given content is connected to a small subset of peers picked at random among all the peers that download that content. In fact, even though peers in the same ISP are downloading the same content they are not necessarily connected to each other. As a consequence, peers unnecessarily download most of the content from peers located outside of their ISP.

Therefore, even if current peer-to-peer content replication solutions significantly reduce content provider costs, they cannot be promoted as a global solution for content replication as they induce huge costs for ISPs. In particular, the current trend for ISPs is to block peer-to-peer traffic [12].

One solution to this problem is to use peer-to-peer locality, that is to constrain peer-to-peer traffic within ISPs' boundaries in order to minimize the amount of inter-ISP traffic.

The seminal work of Karagiannis et al. [14] is the first one to suggest the use of locality in a peer-to-peer system in order to reduce the load on inter-ISP links. They show on real traces the potential for locality (in particular spacial and temporal correlation in the requests for contents) and, based on simulation on a BitTorrent tracker log, they evaluate the benefit of several architectures and in particular a peer-to-peer architecture exploiting locality. More recently, Xie et al. [21] proposed P4P, an architecture to enable cooperation between peer-to-peer applications and ISPs. They show performing large field tests that P4P enables reduction of external traffic for a monitored ISP and enables a reduction on the peers download completion time. Choffnes et al. [8] proposed Ono a BitTorrent extension that leverage on a CDN infrastructure to find localization of peers in order to group peers that are close to each other. They show the benefit of Ono in terms of peers download completion time and suggest using indirect measurements (IP hops and AS hops among peers) that Ono can also reduce inter-ISP traffic.

With those works, there is no doubt about the benefit of peer-to-peer locality and that there are several ways to implement such a locality. However, several fundamental questions are left unanswered by those previous works.

- How far can we push locality? In all proposed solutions the number of inter-ISP connections is kept high enough to guarantee a good robustness. However, this robustness is at the expense of a larger inter-ISP traffic. How far can we push locality without impacting the robustness of the peer-to-peer protocol?

- What is, at the scale of a torrent, the reduction of traffic that can be achieved with locality? Previous works either consider indirect measurements (like the distribution of the number of AS between connected peers), partial measurements (like the monitoring of a specific inter-ISP link), or simulations (like comparing various content distribution scenarios based on the location of peers obtained from a tracker log). However, it is fundamental to understand how the load is spread on all ISPs and what is the reduction per ISP running real experiments.

- Can locality significantly deteriorate the peers experience? Previous works have shown that due to inter-ISP congestion, or due to ISPs offering optical fiber access and unconstrained intra-ISP traffic, locality has the potential to significantly improve peers download completion time. Conversely, in case there is no inter-ISP congestion, can locality degrade the peers download completion time?

The answers to those questions are fundamental if ever peer-to-peer content replication is used by content providers for large scale distribution. In that case, it is likely that the ISPs will request content providers to minimize the traffic due to those peer-to-peer applications on inter-ISP links, but at the same time the content providers will need a clear understanding of the impact of this reduction of traffic on their clients.

Our contribution in this paper is to answer those questions by running extensive large scale BitTorrent experiments (with up to 10 000 real BitTorrent clients) in a controlled environment. In particular, we show that:

- high locality values can save up to two orders of magnitude traffic on inter-ISP links without a negative impact on peer download completion time;
- the capacity of the initial seed is critical to have a low inter-ISP traffic and a low peer download completion time with a high locality;
- models that assume good piece diversity might significantly underestimate the inter-ISP traffic;
- the choice of a high locality values is robust to the torrent size and the number of peers per ISPs;
- in case of congestion on inter-ISP links, high locality values enable a lower peer download completion time than BitTorrent without locality constraint, but in case there is no congestion high locality values increases the peer download completion time;
- high locality values are not robust to churn, because, a fraction of the peers never complete in case of churn, but we identify the reason of this problem and propose a modification of a BitTorrent algorithm to solve this issue and make high locality values also robust to churn.

The remaining of this paper is organized as follow. In section 2, we present an overview of BitTorrent. We define locality and describe our experimental setup, parameters, and metrics in section 3. We discuss the impact of locality in section 4 and the robustness of high locality values in section 5. We describe an extension to the BitTorrent algorithm that solves an issue in the protocol in section 6. In section 7, we discuss the related work. Finally, we conclude in section 8.

2. BITTORRENT OVERVIEW

BitTorrent [9] is a peer-to-peer protocol designed for efficient content distribution. A specificity of BitTorrent is the notion of *torrent* that is a set of peers sharing the same content. A peer can be a *seed*, i.e., a peer with an entire copy of the content, or a *leecher*, i.e., a peer that still needs to download pieces of the content. The first seed in the torrent is called *initial seed*.

To join a torrent, a peer retrieves a *metainfo file*, also called a torrent file, usually from a well known Web site. This file contains the information necessary to download the content, including the IP address and port number of the *tracker* that is the only one centralized component in the BitTorrent architecture. The tracker acts as a *rendez-vous* point for the new peers who want to join a torrent.

Each peer contacts the tracker to get a new list of peers on two events. The first event is when a peer joins the torrent. The tracker then returns a random subset of the peers already in the torrent. A typical number of peers to return is 50, which is what we use for our experiments. The new peers then attempt to initiate connections with some of the peers returned by the tracker. The *peer set* of a peer is formed by all peers to which it has a connection. We call those peers *neighbors*. A peer initiates at most 40 connections and its maximum peer set size is 80. The difference between the number of initiated connections and the maximum peer set size gives the number of incoming connections that can be accepted by this peer. However, different BitTorrent clients may have different peer set sizes, e.g., Vuze (formerly Azureus) clients have 50 neighbors per torrent.

The second event is when the number of neighbors of a peer drops below a given threshold. A peer typically recontacts the tracker in case it has less than 20 connections if it has never received any incoming connections, or less than 7 otherwise. A peer can at most contact the tracker every five minutes, and the tracker can ask the peer to increase this interval in case it is overloaded.

In BitTorrent, a content is split into *pieces* of typically 256kB and each piece is split into *sub-pieces* of 16kB. Pieces are used as the unit of replication, i.e., a peer can only upload a piece for which it has received all sub-pieces. Sub-pieces are the unit of transmission, i.e., peers exchange sub-pieces and perform pipelining [19] on those sub-pieces in order to improve the network utilization.

Once a peer has received its initial peer set from the tracker, it starts downloading the content. Each peer has to answer two main questions: which peers to upload to and which pieces to ask to peers who upload to him. This is the peer and piece selection strategies that are called respectively *choke* and *rarest-first* algorithms in BitTorrent.

The choke algorithm selects the fastest peers to upload to. For a given peer P, the choke algorithm will select 4 peers in the peer set of P to upload pieces to. Three peers are the fastest uploaders to peer P and one peer is selected at random in the peer set of P. The result of this choking algorithm is that it provides an effective sharing incentive, the faster a peer uploads the more it receives, and therefore a high utilization of the upload capacity of peers [15]. However, the fastest peers also need to have pieces to trade with each other.

The rarest-first algorithm on peer P selects the rarest pieces to download from the peers who upload to P. Once the best peers are selected, those peers choose which pieces to download using the rarest first algorithm. Each peer requests pieces that are the rarest in its peer set. This algorithm guarantees a good diversity of the pieces in the

torrent, thus improving the chance that peers always have pieces to trade [16].

3. METHODOLOGY

In this section, we start by defining locality, we then describe our experimental setup, the parameters that we use in our experiments, and finally the metrics that we rely on to evaluate our experiments.

3.1 Locality

We define locality as the percentage of intra-ISP connections over all connections, that is the percentage of traffic constrained within an ISP. Consider a locality of l% and peer P_A from an ISP A that asks the tracker a set of peers. For each peer returned by the tracker to P_A, the tracker will pick, with a uniform distribution, a random number in $[1, 100]$. The tracker will return to P_A a random peer within ISP A if this number is lower than l, and outside ISP A otherwise. Therefore, for a locality of 10%, the tracker returns to P_A neighbors within A with probability 0.1 and outside A with probability 0.9. In the following, we refer to *locality policy* when the tracker implements this notion of locality. We note that to implement our locality policy, we only need to modify the tracker, but not the client.

BitTorrent corresponds to a given, typically low, locality. With BitTorrent, each peer has a connection to a random set of peers. Thus, there is a small percentage of intra-ISP connections. Moreover, with peers uniformly distributed across n ISPs, BitTorrent is equivalent to a scenario with a locality of $\frac{100}{n}$% because if we choose an ISP and pick a random peer, that peer has $\frac{100}{n}$% of chances to be located inside the chosen ISP. In the following, we refer to *BitTorrent policy* when the tracker does not implement our locality policy, but the regular random policy.

With locality, the tracker applies the locality policy to all peers except the initial seed. Because the goal of the initial seed is to improve diversity, the tracker selects the neighbors of the initial seed using the BitTorrent policy. However, we apply the locality policy to all the other seeds, that is all the leechers that become seed during the experiments. Note that the traffic generated by the initial seed is negligible compared to the aggregated traffic of the torrent.

3.2 Experimental Setup

In this paper, we have run large scale experiments to evaluate the impact of locality on inter-ISP traffic and BitTorrent download completion time. We have run experiments instead of simulations for two main reasons. First, it is hard to run realistic (packet level discrete) peer-to-peer simulations with more than a few thousand of peers due to the large state generated by each peer and the packets in transit on the links. Moreover, at that scale, simulations are often slower than real time. Second, the dynamics of BitTorrent is subtle and not yet deeply understood. Running simulations with a simplified version of BitTorrent may hide fundamental properties of the system.

As we will see in section 4 and section 5, our results indeed depend on the dynamics at the packet level and on the large scale achieved, up to 10 000 peers.

We now describe the experimental platform on which we run all our experiments, the BitTorrent client that we use in our experiments, and how we simulate an inter-ISP topology atop a computational Grid.

3.2.1 Platform

We obtain all our results by running large scale experiments with a real BitTorrent client.

We run all our experiments on the Grid'5000 experimental platform. A typical node in Grid'5000 has bi or quad-core AMD Opteron CPU, 2 to 4GB of memory, and a gigabit Ethernet connectivity. Grid'5000 consists of 9 sites spread across France and counts 1 700 nodes in total [3]. Once a set of nodes is reserved, no other experiment can run on parallel on those nodes. In particular, there is no virtualization on those nodes. Therefore, experiments are totally controlled and reproducible.

The BitTorrent client used for our experiments is an instrumented version of the mainline client [4], which is based on version 4.0.2 of the official client [2]. This instrumented client can log specific messages received and sent. Unless specified otherwise, we use the default parameters of this client. In particular, each peer uploads at 20kB/s to 4 other peers, and the maximum peer set size is 80. We will vary the upload capacity when studying the impact of heterogeneous upload capacity on locality in section 4 (see section 3.3.1 for a description of our heterogeneous scenario). We also use the choke algorithm in seed state of the official client in its version 4.0.2. This algorithm is somewhat different, as it is fairer and more robust than the one implemented in most BitTorrent clients today [15, 16]. However, as it only impacts the seed, we do not believe this algorithm to have a significant impact on our results. The interested reader can find a detailed description of this algorithm in [15].

We use the following default parameters for our experiments, unless otherwise specified. Peers share a content of 100MB that is split into pieces of 256kB. By default, all peers including the initial seed start within the first 60 seconds of the experiments. However, we will also vary this parameter in section 5.4 when studying the impact of churn on BitTorrent locality (see section 3.2.2 for a description of our scenario with churn). Once they have completed the download, leechers stay 5 minutes as seed and then they leave the torrent. We have chosen 5 minutes seeding time because for smaller values, we have observed that some leechers do not complete. This is due to an issue in the BitTorrent protocol on which we will come back in section 5.4. However, we argue that it should not impact our results because 5 minutes is small compared to the optimal download completion time (83 minutes). The initial seed stays connected for the entire duration of the experiment.

We run all our experiments with up to 100 BitTorrent clients per physical node. Therefore, for torrents with 100, 1 000, and 10 000 peers, we use respectively 1, 10, and 100 nodes for the leechers, plus one node for the seed and the tracker. Each client on a same node uses a different port to allow communication among those clients. We have performed a benchmarking test to find how many clients we can run on a single node without a performance penalty that we identify with a decrease in the client download time for a reference content of 100MB. We have found that we can run up to 150 clients uploading at 20KB/s on a single node with-

out performance penalty. With a larger number of clients, we observe a significant slowdown in the download completion time due to an overload of the node's hard drive. To be safe, we run no more than 100 clients uploading at 20kB/s on one node, or 2MB/s of BitTorrent workload. The BitTorrent workload is aggressive for the hard drive because each client triggers many read and write operations that cannot be cached, thus resulting in many seek operations. When we will vary the upload capacity of clients in section 4, we will then adapt the number of clients per node so that the aggregated upload capacity per node is never beyond 2MB/s.

3.2.2 Inter-ISPs Topology

We remind that our goal is to evaluate the impact of locality on inter-ISP traffic and peers performance. Therefore, we simulated an inter-ISP topology on top of the grid we use to run our experiments. We explain in the following how we simulated this topology and how representative it is of the real Internet.

For all our experiments, we assume that we have a set of stub-ISPs that can communicate among each other. On top of this topology, we consider two scenarios. The first scenario is when all stub-ISPs have a single *peering* link to each other, thus a full mesh. We refer to a peering link as a link for which an ISP does not pay for traffic. However, the peering technology is expensive to upgrade so ISPs are interested in offloading those links. The second scenario is when each stub-ISP is connected with a *transit* link to a *single* transit-ISP. All peers are in stub-ISPs. Therefore, there is no traffic with a source or a destination in the transit-ISP. We refer to a transit link as a link on which traffic is billed according to the 95-th percentile. Therefore, ISPs are interested in reducing the bursts of traffic on those links.

We stress that both scenarios are simply a different interpretation of a same experiment, as all peers are in stub-ISPs and the traffic flows from one stub-ISP to another one. In the following, we refer to *inter-ISP link* when our discussion applies to both peering and transit links.

In our experiments, the notion of ISPs and inter-ISP links is virtual, as we run all our experiments on a Grid. To simulate the presence of a peer in a given ISP, for each experiment, we create a static mapping between peers and ISPs. We use this mapping to compute offline the traffic that is uploaded on each inter-ISP link of the stub-ISPs. For instance, imagine that peer P_A is mapped to the ISP A and peer P_B is mapped to the ISP B. All the traffic sent from P_A to P_B is considered as traffic uploaded by the ISP A to the ISP B with a peering link in the first scenario or with a transit link via the transit ISP in the second scenario.

Our experiments are equivalent to what we would have obtained in the Internet with real ISPs and inter-ISP links except for latency. We argue that latency would not significantly change our results because the choking algorithm is insensitive to latency by design. Indeed, BitTorrent computes the throughput over a 20 seconds sliding window, which should alleviate the impact of latency on TCP ramp up [9].

We experiment with and without bottlenecks in the network. By default, there is no bottleneck in the network because the aggregated traffic generated by our experiments is always significantly lower than the bottleneck capacity of the Grid. However, we also create artificial bottlenecks on inter-ISP links to evaluate their impact on inter-ISP traffic and performances (see section 3.3.2 for the description of how we limit the inter-ISP link capacity). It is important to experiment the impact of bottlenecks on inter-ISP links because the choking algorithm selects peers according to their throughput. Therefore, bottlenecks may significantly change BitTorrent's behavior.

Finally, we have not considered a hierarchy of transit-ISPs. We show in section 4 and section 5, that with high locality the inter-ISP traffic is so low that optimizing this traffic to minimize, for instance, the number of AS hops will bring little benefits.

3.3 Experimental Parameters

This section describes the parameters for the two series of experiments of this paper. The first series experiments with the impact of our locality policy on inter-ISP traffic and peers performance. In the second series, we evaluate the robustness of high locality values to the torrent size, the number of peers per ISP, the inter-ISP link capacity, and the churn rate.

3.3.1 Impact of Locality

For the first series of experiments, we set the torrent size to 1 000 peers and the number of ISPs to 10. Therefore, there are 100 peers per ISP in all the experiments of the first series. To analyze the impact of locality on BitTorrent, we then vary the locality between 10% and 90% by steps of 10% and we also consider the following high locality values: 99%, 99.5%, and 99.9%. We then run experiments for each of the three following scenarios.

Homogeneous scenario with a slow seed In this scenario both the initial seed and the leechers can upload at a maximum of 20kB/s. As we have mentioned earlier, we run 100 leechers per node, and we run the initial seed and the tracker on an additional node. According to the definition of locality from section 3.1, each peer has the same probability to have a connection to the initial seed, whichever ISP it belongs to. For instance, as the initial seed has a peer set of 80, with 10 ISPs, each ISP has in average 8 peers with a connection to this initial seed.

Heterogeneous scenario We experiment with leechers with heterogeneous upload capacities and a fast initial seed. In each ISP, one third of the peers uploads at 20kB/s, one third uploads at 50kB/s, and one third uploads at 100kB/s. For simplicity, we run all the leechers with the same upload capacity on the same node. Because we have determined that the hard drives cannot sustain a workload higher than 2MB/s, we run only 20 clients per node. For BitTorrent to perform optimally, the initial seed uploads at 100kB/s, as fast as the the fastest leechers [15]. Each ISP has the same probability to have a connection to the initial seed, whichever ISP it belongs to.

We experiment with heterogeneous upload capacities for three reasons. The first reason is that non-local peers may be faster than local peers so the local peers

may unchoke inter-ISP connections more often than intra-ISP connections, thus making locality inefficient to reduce inter-ISP traffic. The second reason is that local peers may be faster than non-local peers so inter-ISP connections may be rarely used to download new pieces, thus degrading performances. The third reason is that in case of heterogeneous upload capacities inside an ISP, if fast peers are those with the inter-ISP connections, slower peers may not be given pieces to trade among themselves, also degrading performances.

Homogeneous scenario with a fast seed We experiment with leechers that upload at 20kB/s and an initial seed that uploads at 100kB/s. We run this additional experiment in order to understand whether the results obtained with the heterogeneous scenario are due to the fast initial seed or due to the heterogeneous capacities of leechers.

3.3.2 Robustness of High Locality Values

According to our definition of locality, the number of inter-ISP connections for a given locality value depends on the number of peers in the ISPs. To compare results for various torrent sizes and number of peers per ISP, we fix the number of inter-ISP connections to 8 and 80, which corresponds to high locality values.

To evaluate the impact of a given number of inter-ISP connections, we select for each experiment the locality value that gives this number of inter-ISP connections on average. For i inter-ISP connections, m peers per ISP, and 80 neighbors per peer, the corresponding locality value is $100 \cdot (1 - \frac{i}{m \cdot 80})$

In all the experiments that evaluate the robustness of high locality values, for 8 inter-ISP connections the locality values range from 99% (for 100 peers and 10 ISPs) to 99.998% (for 10 000 peers and 5 000 peers per ISP), and for 80 inter-ISP connections, the locality values range from 90% (for 100 peers and 10 ISPs) to 99.98% (for 10 000 peers and 5 000 peers per ISP).

For the second series of experiments, we consider a scenario with homogeneous upload capacities, a slow seed, and 8 or 80 inter-ISP connections. Then, we vary the torrent size, the number of peers per ISP, the inter-ISP link capacity, and the churn rate. Note that we vary only one parameter at a time per experiment.

Torrent size We consider torrents with 100, 1 000, and 10 000 peers with 10 ISPs.

Number of peers per ISP We evaluate 100, 1 000 and 5 000 peers per ISP. To vary the number of peers per ISP, we vary the number of ISPs with a constant torrent size of 10 000 peers. Therefore, to obtain 100, 1 000 and 5 000 peers per ISP, we consider 100, 10, and 2 ISPs. Note that we also experiment with as few as 10 peers per ISP in the scenario with a torrent size of 100 peers and 10 ISPs.

Inter-ISP link capacity We vary the inter-ISP link capacity from 40kB/s to 2 000kB/s. However, local peers can upload to their local neighbors (in the same ISP) at 20kB/s without any limitation. For this experiment, all the BitTorrent clients that run on the same node are located in the same virtual ISP, so limiting the upload capacity of the node is equivalent to limiting that inter-ISP link capacity. For an inter-ISP link capacity of 2 000kB/s, all the BitTorrent clients that are located on a same node can upload outside this ISP at their full capacity without any congestion. Therefore, it is equivalent to the case with no inter-ISP link bottleneck. We use the tool traffic controller (tc), that is part of the iproute2 package, to limit the upload capacity of each node on which we run experiments. We deploy our own image of GNU/Linux, on which we have superuser privileges, on all the nodes we want to limit the upload capacity on. Limiting the upload capacity on each node allows us to reproduce Internet's bottlenecks in a controlled environment.

Churn We start a first set of 10 000 peers uniformly within the first 60 seconds in a first experiment, 600 seconds in a second experiment, and 6 000 seconds in a third experiment. Then, when each of those peers complete its download, we start a new peer from a second set of 10 000 peers. Hence, we model the three phases of a real torrent's life: flashcrowd, steady phase, and end phase [13]. The first phase, the flashcrowd, occurs while all peers of the first set join the torrent. The second phase, the steady phase, occurs when the number of peers in the torrent remains constant. This is when peers in the first set start to complete and that peers in the second set are started to replace those peers in order to keep the torrent size constant to 10 000 peers. The last phase, the end phase, occurs at the end of the torrent's life, when the last peers complete their download and no new peer joins the torrent. This is when there is no more peers in the second set to compensate departure of peers.

3.4 Evaluation Metrics

To evaluate our experiments, we consider three metrics: the content replication overhead, the 95th percentile, and the peer slowdown.

Overhead For each stub-ISP, we monitor the number of pieces that are uploaded from this stub-ISP to any other stub-ISP during the experiment. Then, to obtain the per-ISP content replication overhead, we normalize the amount of data uploaded by the size of the content for the experiment. Thus, we obtain the overhead in unit of content that crosses an inter-ISP link. We call this metric the content replication overhead, or overhead for short, because with the client-server paradigm, ISPs would upload zero byte. We use the overhead as a measure of load on peering links.

95th Percentile To obtain the 95th percentile of the overhead, we compute the overhead by periods of 5 minutes and then consider the 5 minutes overhead corresponding to the 95th percentile. The 95th percentile is the most popular charging model used on the Internet [17].

Slowdown We define the ideal completion time of a peer as the time for this peer to download the content at a

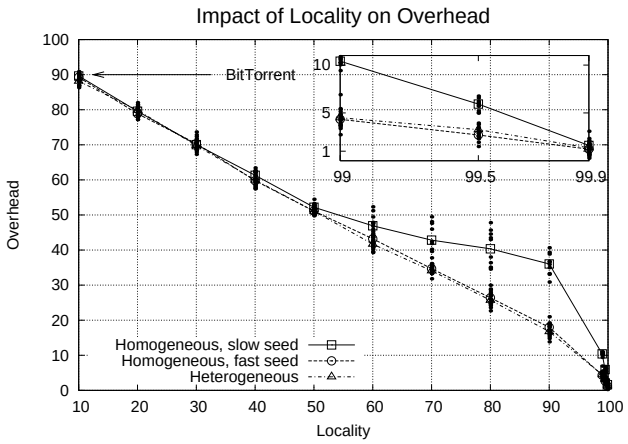

Figure 1: Overhead with 1 000 peers and 10 ISPs for various values of locality. Each square, circle and triangle represents the average number of copies uploaded by an ISP for a particular value of locality in a given scenario. Each dot represents this overhead for one ISP. *We can reduce BitTorrent overhead by up to two orders of magnitude.*

speed equivalent to the average of the maximum upload capacity of all peers. This is the best completion time, averaged over all peers, that can be achieved in a peer-to-peer system in which each peer always uploads at its maximum upload capacity. The slowdown is the experimental peer download completion time normalized by the ideal completion time. For instance, imagine that all peers have the same maximum upload capacity of 20kB/s. An average peer slowdown of 1 for 10 000 peers means that there is an optimal utilization of the peers upload capacity, or that the peers are, on average, as fast as a client-server scenario in which we have 10 000 servers, one server per client sending at 20kB/s.

4. IMPACT OF LOCALITY

The goal of this section is to explore the relation of locality with the overhead and slowdown considering a large range of locality values. In particular, we will evaluate how far we can push locality to obtain the smaller overhead attainable without impacting adversely the slowdown.

We make experiments with 1 000 peers and 10 ISPs in three scenarios: homogeneous with a slow seed, heterogeneous with a fast seed, and homogeneous with a fast seed. First, we evaluate the impact of locality on overhead and 95th percentile. Then, we evaluate the impact of locality on slowdown.

4.1 Impact on Overhead

In this section, we will discuss three main results.

- The overhead decreases linearly with locality. The observed gain compared to BitTorrent is up to two orders of magnitude for high locality values.
- The initial seed upload capacity has a critical impact on the overhead. In particular, with an insufficient initial seed provisioning, the overhead is surprisingly higher than with a well provisioned initial seed.
- The 95th percentile is larger for the heterogeneous scenario than for the two other ones.

We observe in Fig. 1 that for the two scenarios with a well provisioned initial seed, i.e., the homogeneous fast seed and the heterogeneous scenarios, the overhead decreases linearly with locality. We explain this behavior because a well provisioned initial seed guarantees a high level of piece diversity in each ISP. Indeed, we remind that with the locality policy described in section 3.1, the initial seed is the only one peer to be returned at random with a uniform distribution to other peers. Thus, as the initial seed has 80 connections and there are 10 ISPs, this initial seed has on average 8 connections per ISP. Therefore, as there is a high piece diversity, for a locality of l%, a peer in a given ISP has a probability of $(100 - l)$% to unchoke peers outside his own ISP, thus the linear relation between overhead and locality. Indeed, when there is no congestion in the network and a uniform repartition of the upload capacity of peers in each ISP, the probability to unchoke a peer outside his own ISP is only determined by the number of neighbors this peer has outside his own ISP. We evaluate the impact of network bottlenecks in section 5.3.

For high locality values, that is values larger than 99%, Fig. 1 shows that the overhead still decreases linearly but with a larger negative slop, that is the overhead decreases faster for locality values larger than 99% than for smaller ones. As for the case with lower locality values, increasing the locality decreases linearly the opportunity to unchoke peers outside a given ISP. However, with high locality values, as there are few inter-ISP connections, those connections are used all the time, unlike the case with lower locality values in which the inter-ISP connections are not always used. As a consequence, reducing by a given amount the number of inter-ISP connections has a larger impact on overhead when those connections are all used. Thus, this explain the larger negative slop for high locality values.

The BitTorrent arrows in Fig. 1 and Fig. 2 represents the value of respectively overhead and 95th percentile achieved by BitTorrent in the same scenario. Indeed, with 1 000 peers and 10 ISPs of 100 peers, each peer has 10% of connections inside his own ISP with the BitTorrent policy. Therefore, BitTorrent corresponds to the case with 10% of locality in our experiments.

For all three scenarios, our locality policy allows to save up to two orders of magnitude of bandwidth on inter-ISP links. Indeed, we see in Fig. 1 that for a locality of 10%, the case of the BitTorrent policy, the overhead is close to 90, and for a locality of 99% the overhead is lower than 10 for all three scenarios. For the highest locality value considered, 99.9%, the overhead is close to one for all three scenarios.

Surprisingly, we observe that for locality values between 50% and 90% there is a higher overhead for the homogeneous scenario with a slow seed than for the two other scenarios with a fast seed, see Fig. 1 . As we explained, due to the high piece diversity, there is a linear relation between overhead and locality. However, for locality values large enough, reducing the number of connections among ISPs will impact piece diversity. As a result, peers in a given ISP will have

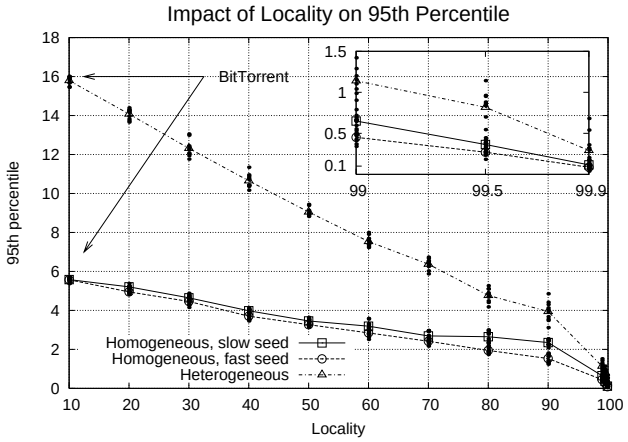

Figure 2: 95th percentile with 1 000 peers and 10 ISPs for various values of locality. Each square, circle and triangle represents the average maximum number of copies uploaded by an ISP during a 5 minutes period. Each dot represents this 95th percentile for one ISP. *Heterogeneity leads to a higher 95th percentile than homogeneous ones, but high locality values achieve up to two orders of magnitude reduction of this 95th percentile for all scenarios.*

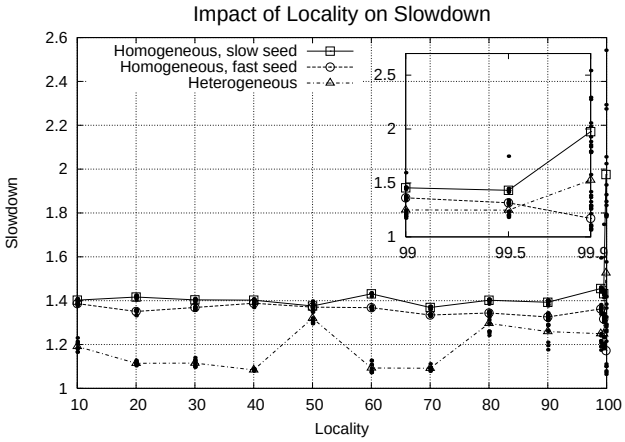

Figure 3: Slowdown with 1 000 peers and 10 ISPs for various values of locality. Each square, circle and triangle represents the slowdown of the average completion time of peers as compared to the best achievable completion time for a particular value of locality in a given scenario. Each dot represents this slowdown for one ISP. *The locality does not impact peers slowdown.*

to use more their inter-ISP connections, thus a higher overhead, in order to download pieces that are missing in their own ISP. We do not observe the same issue with the two scenarios with a fast seed because this fast initial seed is fast enough to guarantee a high piece diversity even for low locality values.

This issue of piece diversity increasing the overhead when the initial seed is not fast enough shows that simple models that evaluate the overhead based on the assumption that there is a good piece diversity might significantly *underestimate* the real overhead. Unless very high values of locality are used, larger than 99% in our experiments, the upload capacity of the initial seed is critical to the overhead.

We also observe a linear relation between locality and 95th percentile as well as a significant reduction of the 95th percentile for high locality values in Fig. 2. However, we observe that the 95th percentile for the heterogeneous scenario is much larger than for the two other scenarios. This is because in the heterogeneous scenario there are two third of the peers that are faster than 20 kB/s, which is the upload capacity of all the peers for the two other scenarios. Therefore, we see that even if the total amount of traffic crossing inter-ISP links is not significantly impacted by the distribution of the upload capacity of peers (see Fig. 1), this distribution might have a major impact on the 95th percentile that is used for charging traffic on transit links.

In summary, we have seen that the relation between the overhead (or 95th percentile) and locality is not trivial, but depends on the dynamics of the BitTorrent protocol, and in particular on the initial seed upload capacity and peers upload capacity. We have also shown that very high locality values, larger than 99%, lead to a major reduction of the overhead and 95th percentile up to two order of magnitude. Also, models that assume good piece diversity might significantly underestimate the overhead. In the next section, we explore what is the impact of such a high reduction on the peers slowdown.

4.2 Impact on Slowdown

Fig. 3 shows that the locality value has no impact on peers slowdown for the two scenarios with a fast seed (heterogeneous and homogeneous with a fast seed) and no impact for a locality value up to 99.5% for the homogeneous scenario with a slow seed. This result is remarkable when one consider the huge saving the high locality values enable for the overhead and 95th percentile. This result shows the high efficiency BitTorrent achieves even when the peers interconnectivity graph is highly constrained.

For a locality of 99.9% with the homogeneous scenario with a slow seed, the slowdown increases by 43% compared to lower locality values. This increase is due to a poor piece diversity, which can be avoided by having a fast initial seed as shown by the two scenarios with a fast seed in Fig. 3. Moreover, even if a 43% increase is not negligible, it has to be compared with the huge saving achieved by a locality value of 99.9%. Indeed, in Fig. 1 we see for the homogeneous scenario with a slow seed that even if the overhead is already very low (close to 6) for a locality value of 99.5%, it is close of 1 for a locality of 99.9%.

We observe that the slowdown for the heterogeneous scenario varies between 1.1 and 1.4 in Fig. 3. Those variations are due to the random distribution of the connections among peers with different upload capacities. Whereas with the homogeneous scenarios the random selection of neighbors does not impact significantly the slowdown because all neighbors have the same upload capacity, with the heterogeneous scenario, small variations in the peer set of peers will significantly impact the slowdown. This is in particular the case

when fast peers are connected to a higher proportion of slow peers. In that case, the fast peers will need slightly longer to complete, therefore they will contribute longer, which is to the benefit of other peers. As the fast peers are much faster than the other ones, small variations lead to noticeable change in the slowdown.

Our last observation on Fig. 3 is that for most scenarios and locality values, the slowdown is no more than 1.4. According to our definition of the slowdown, BitTorrent is only 40% slower than a system that is able to serve all peers with their aggregate upload capacity for the entire duration of the experiment. This result shows that high efficiency of BitTorrent, which will be confirmed in section 5.1.2 for torrents with 10 000 peers.

In conclusion, we see that the peers slowdown is not impacted by the locality value even for high values.

4.3 Discussion

We have seen that a high locality value dramatically reduces the overhead and 95th percentile without any impact on peers slowdown. In particular, we have seen that we can consider very high locality values and that the only one limiting factor that can impact peers slowdown is the upload capacity of the initial seed.

Indeed, the initial seed upload capacity is critical to the high piece diversity in a torrent in its flashcrowd phase, confirming findings of Legout et al.[15]. As the overhead and peers slowdown are impacted by this piece diversity, the choice of the locality value might be hard in practice because the characteristics of the torrent may impact this diversity.

In the following, we evaluate the robustness of high locality values to the characteristics of the torrent (torrent size, number of peers per ISP, churn rate), or the characteristics of the network (limitation of the capacity of the inter-ISP links).

5. ROBUSTNESS OF HIGH LOCALITY VALUES

In this section, we evaluate the impact of a 8 and 80 inter-ISP connections on average on the overhead and slowdown when we vary the torrent size, the number of peers per ISP, the inter-ISP link capacity, and the churn rate.

In the following, we do not present results for the 95th percentile, as they do not show any significant new insights compared to the results for the overhead.

5.1 Impact of the Torrent Size

We consider torrents with 100, 1 000, and 10 000 peers. For each torrent we set the number of ISPs to 10. In the following, we evaluate the overhead and peers slowdown for those torrents with 8 and 80 inter-ISP connections.

5.1.1 Impact on Overhead

In Fig. 4, we see that for a small number of inter-ISP connections the overhead is low and almost independent of the torrent size, whereas for the BitTorrent policy it increases linearly with the torrent size.

For the torrent with 100 peers, as there are 10 ISPs, there are only 10 peers per ISP. This scenario is interesting because a locality policy only makes sense when there are

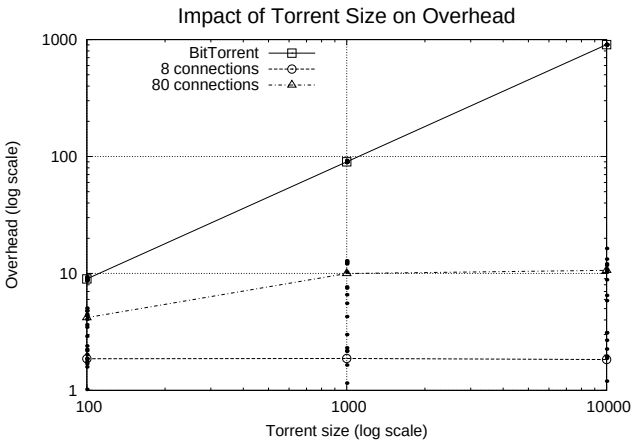

Figure 4: Overhead of torrents with 100, 1 000 and 10 000 peers and 10 ISPs in three scenarios: BitTorrent policy, locality policy with 8 inter-ISP connections and 80 inter-ISP connections. Each square, circle and triangle represents the average number of copies uploaded by an ISP for a particular torrent size in a given scenario. Each dot represents this overhead for one ISP. *With high locality values, the overhead remains low and almost constant for all torrent sizes.*

enough peers inside each ISP to be able to keep traffic local. This scenario shows the gain that can be achieved for a low number of peers per ISP. Due to an artifact in the way the tracker assigns neighbors with the locality policy, the effective number of inter-ISP connections for this torrent of 100 peers is on average 0.9[1] and 9 instead of respectively 8 and 80. Therefore, with this lower effective number of inter-ISP connections, the overhead is slightly lower and the slowdown slightly higher than what it should be for this torrent. However, we do not believe this artifact to significantly change our results, and we will correct it in future work. Moreover, this artifact does not arise in any other experiment we torrents larger than 100 peers.

With a torrent of 100 peers, we save 80% of overhead as compared to BitTorrent with 0.9 inter-ISP connections and 54% with 9 inter-ISP connections. With a torrent of 10 000 peers, we save 99.8% of overhead as compared to BitTorrent with 8 inter-ISP connections and 98% with 80 inter-ISP connections.

In summary, we have seen that the choice of the locality value is robust to the torrent size. Indeed, a high locality value keeps the overhead low for all the torrent sizes we have considered.

5.1.2 Impact on Slowdown

For all the considered torrent sizes and 8 inter-ISP connections, the slowdown is higher than BitTorrent but for 80 inter-ISP connections, it is close to BitTorrent (see Fig. 5).

We see in Fig. 5 that with 8 inter-ISP connections the

[1]An average number of inter-ISP connections of 0.9 means that some ISPs have 0 zero connection to other peers. However, as the initial seed is not subject to the locality policy, it has connection to all ISPs. Thus, all peers in all ISPs are able to complete the download.

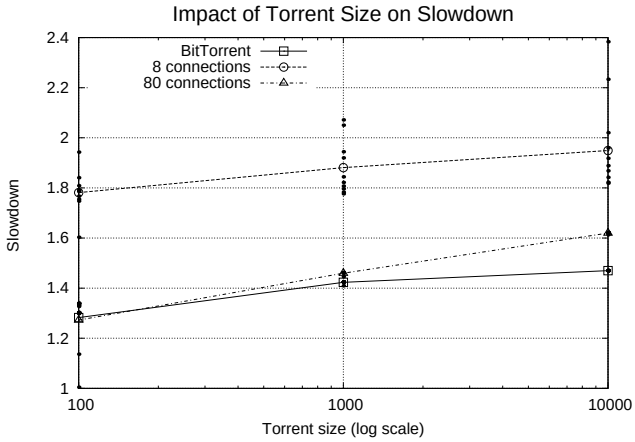

Figure 5: Slowdown of torrents with 100, 1 000 **and** 10 000 **peers and** 10 **ISPs in three scenarios: BitTorrent policy, locality policy with** 8 **inter-ISP connections and** 80 **inter-ISP connections. Each square, circle and triangle represents the slowdown of the average completion time of peers as compared to the best achievable completion time. Each dot represents this slowdown for one ISP.** *High locality values lead to a negligible to moderate increase in peers slowdown.*

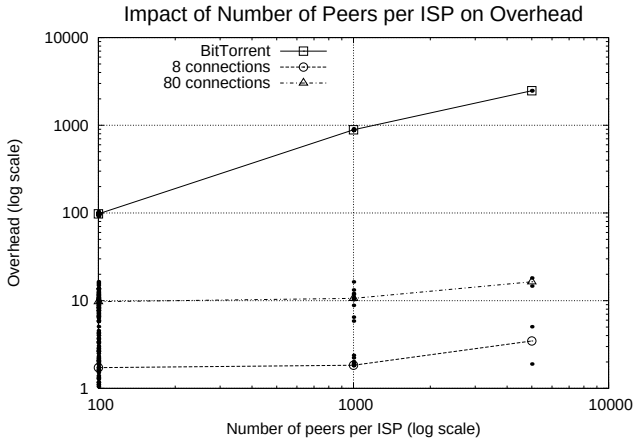

Figure 6: Overhead with 10 000 **peers and** 100, 1 000, **and** 5 000 **peers per ISP for three scenarios: BitTorrent policy, locality policy with** 8 **inter-ISP connections and** 80 **inter-ISP connections. Each square, circle and triangle represents the average number of copies uploaded by an ISP for a particular number of peers per ISP in a given scenario. Each dot represents this overhead for one ISP.** *For high locality values, the overhead increases slowly with the number of peers per ISP.*

slowdown is 38% higher than with the BitTorrent policy for a torrent with 100 peers. We remind that a torrent with 100 peers and 10 ISPs each ISP has 0.9 connection on average. For 1 000 and 10 000 peers, the slowdown is 32% higher than with the BitTorrent policy

With 80 inter-ISP connections, the slowdown is equal to the one achieved by the BitTorrent policy for 100 peers, 2% higher than with the BitTorrent policy for 1 000 peers, and 10% higher than with the BitTorrent policy for 10 000 peers.

As a side note, we confirm, according to Fig. 5, the high efficiency of BitTorrent. We observe that for the BitTorrent policy, the slowdown increases sub-logarithmically with the torrent size, and that for 10 000 peers, the slowdown is lower than 1.5, which is quite remarkable.

In summary, the choice of the locality value is robust to the torrent size when one consider the peers slowdown metric. Indeed, for a given locality value the slowdown increases logarithmically with the torrent size, as shown by the linear trend of the slowdown with a log scale in Fig. 5.

We have seen that 80 inter-ISP connections lead to a significant overhead saving and to a performance close to the one achieved by the BitTorrent policy for a large range of torrent sizes. However, in case a higher overhead saving is required, 8 inter-ISP connections lead to an overhead close to the lowest attainable overhead using BitTorrent (that is, on average, only one copy of the content uploaded per-ISP) at the expense of a moderate increase in the slowdown in the order of 30%.

5.2 Impact of the Number of Peers per ISP

In this section, we perform experiments with torrents of 10 000 peers and we vary the number of peers per ISP considering 100, 1 000, and 5 000 peers per ISP. As we keep the torrent size constant, we vary the number of peers per ISP by considering scenarios with 100, 10, or 2 ISPs.

We evaluate the impact of the number of peers per ISP for 8 and 80 inter-ISP connections on the overhead and slowdown.

5.2.1 Impact on Overhead

Fig. 6 shows that with a fixed number of inter-ISP connections, the overhead increases slowly with the number of peers per ISP. For 8 inter-ISP connections, the overhead is 1.7 for 100 peers per ISP, 1.8 for 1 000 peers per ISP, and 3.4 for 5 000 peers per ISP. Hence, an ISP with 5 000 peers only needs to download 1.7 additional copy of the content compared to an ISP with 100 peers for all peers to complete. The overhead is also small as compared to BitTorrent. For any number of peers per ISP, the scenario with 8 inter-ISP connections saves at least 99% of overhead as compared to BitTorrent.

In summary, the locality value is robust to the number of peers per-ISP when on consider the overhead metric.

5.2.2 Impact on Slowdown

We see in Fig. 7 that for 100, 1 000, and 5 000 peers per ISP, the slowdown is higher than BitTorrent for 8 inter-ISP connections but it is close to BitTorrent for 80 inter-ISP connections.

Indeed, with 8 inter-ISP connections and 100 peers per ISP, the slowdown is 58% higher than BitTorrent and the maximum slowdown over all ISPs is more than twice as high as the minimum slowdown. This is because with 8 inter-ISP connections, the diameter of the overlay significantly increases as we increase the number of ISPs. This higher diameter explains the heterogeneity observed per-ISP slowdown (shown by the dots in Fig. 7). With 80 inter-ISP

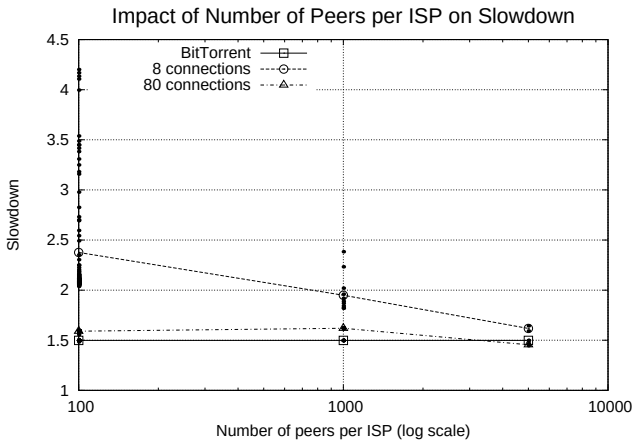

Figure 7: Slowdown with 10 000 peers and 100, 1 000, and 5 000 peers per ISP for three scenarios: BitTorrent policy, locality policy with 8 inter-ISP connections and 80 inter-ISP connections. Each square, circle and triangle represents the slowdown of the average completion time of peers as compared to the best achievable completion time for a particular number of peers per ISP in a given scenario. Each dot represents this slowdown per ISP *The slowdown with locality is close to the one of the BitTorrent policy except for 8 inter-ISP connections and a 100 peers per ISP.*

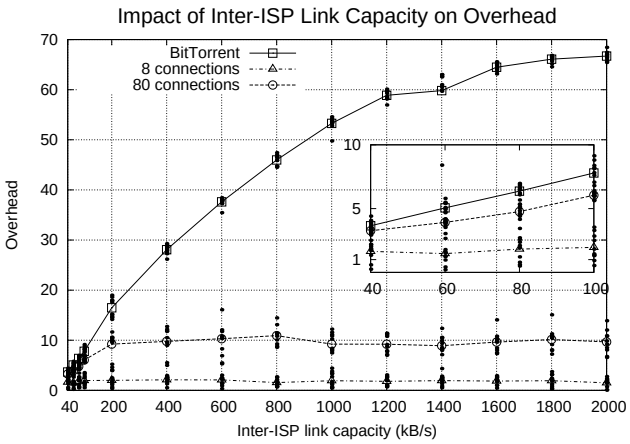

Figure 8: Overhead with 1 000 peers and 10 ISP for various inter-ISP link capacities and three scenarios: BitTorrent policy, locality policy with 8 inter-ISP connections and 80 inter-ISP connections. Each square, circle and triangle represents the average number of copies uploaded by an ISP for a particular inter-ISP link capacity in a given scenario. Each dot represents this overhead for one ISP. *The overhead is significantly reduced with locality for any inter-ISP link capacity.*

connections though, the slowdown is only 6% higher than BitTorrent in the worst case.

In summary, the locality value is robust to the number of peers per ISP in most case, but 8 inter-ISP connections leads to a slowdown around 60% higher than the one achieved by the BitTorrent policy for a torrent of 100 peers.

We have seen that for various number of peers per ISP, 80 inter-ISP connections lead to the good tradeoff between low overhead and a slowdown close to the one achieved by the BitTorrent policy. However, 8 inter-ISP connections lead to an overhead close to the minimum one, but at the expense of a higher slowdown, in particular for 100 peers per ISP. We believe this result to show that high locality values are robust to the number of peers per ISP because even for very challenging scenarios (torrent with 100 peers and 8 inter-ISP connections) the overhead is significantly reduced and the increase in slowdown is low enough to be reasonable for a content provider.

5.3 Impact of the Inter-ISP Link Capacity

In this section, we vary the inter-ISP link capacity from 40kB/s to 100kB/s by steps of 20kB/s and from 200kB/s to 2 000kB/s by steps of 200kB/s. We first evaluate the impact of the inter-ISP link capacity on the overhead. Then, we evaluate the impact of the inter-ISP link capacity on the slowdown.

5.3.1 Impact on Overhead

We see in Fig. 8 that with high locality values the overhead remains low. Indeed, for 8 inter-ISP connections, the overhead is close to 1.5 for any inter-ISP link capacity. For 80 inter-ISP connections the overhead is close to 10 for any inter-ISP link capacity, except for a capacity lower than 200 kB/s. For the BitTorrent policy, the overhead increases with the inter-ISP link capacity. The reason is that BitTorrent, due to the choke algorithm, will prefer to exchange data with local peers when there is congestion on the inter-ISP links, because those local peers are not on a congested path, thus a larger BitTorrent download speed. For high inter-ISP link capacity, those links are no more congested, therefore the capacity does not impact anymore the overhead achieved by the BitTorrent policy.

In summary, the overhead is almost independent of the inter-ISP link capacity for high locality values, whereas it significantly increases with the inter-ISP link capacity for the BitTorrent policy. Moreover, the locality policy results in high overhead saving compared to the case of the BitTorrent policy.

5.3.2 Impact on Slowdown

We observe in Fig. 9 that with congestion on inter-ISP links, high locality values improve the peers slowdown. Indeed, the scenario with the BitTorrent policy becomes slower than the scenario with 8 inter-ISP connections for an inter-ISP link capacity lower than 800 kB/s, and slower than the scenario with 80 inter-ISP connections for an inter-ISP link capacity lower than 400 kB/s. The benefit of high locality values on the slowdown is significant for highly congestion inter-ISP links. Indeed, for an inter-ISP link capacity of 40kB/s, the scenario with 8 inter-ISP connections is 200% faster than with the BitTorrent policy.

In summary, high locality values can significantly improve the peers slowdown in case there is congestion or capacity limitation on the inter-ISP links.

We have seen that high locality values bound the overhead

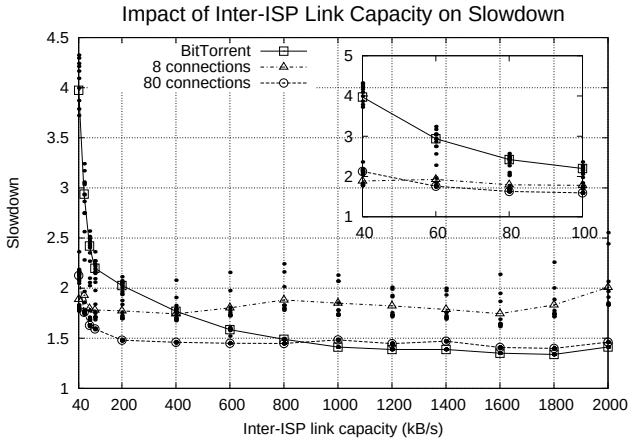

Figure 9: Slowdown with 1 000 peers and 10 ISPs for various inter-ISP link capacities and three scenarios: BitTorrent policy, locality policy with 8 inter-ISP connections and 80 inter-ISP connections. Each square, circle and triangle represents the slowdown of the average completion time of peers as compared to the best achievable completion time for a particular inter-ISP link capacity in a given scenario. Each dot represents this slowdown for one ISP. *With congested inter-ISP links, high locality values improve the slowdown compared to the case of the BitTorrent policy.*

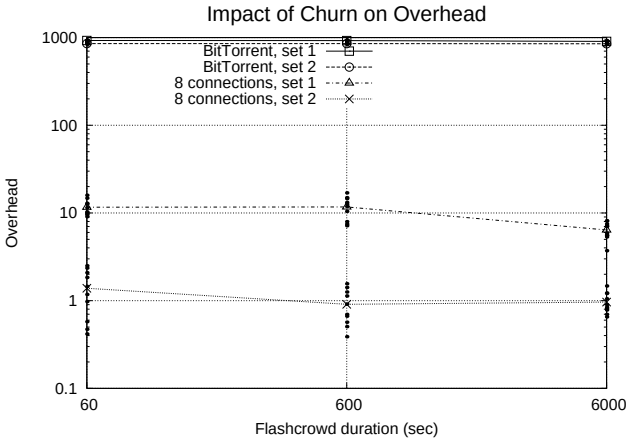

Figure 10: Overhead with a torrent of 10 000 peers and 10 ISPs with churn. All peers in set 1 start with a uniform distribution within the specified flashcrowd duration. Each peer in set 2 starts when a peer from set 1 has completed its download. Set 1 and 2 consists of 10 000 peers, but there are no more than 10 000 simultaneous peers in the torrent. We consider two scenarios: BitTorrent policy and locality policy with 8 inter-ISP connections. The lines represent the average overhead for all ISPs. Each dot represents this overhead for one ISP. *The churn does not adversely affect the overhead with a high locality value.*

independently of the inter-ISP link capacity, but also significantly improves the peers slowdown in case of inter-ISP links congestion. Therefore, high locality values are robust to inter-ISP congestion.

5.4 Impact of the Churn Rate

In this section, we evaluate the impact of churn with 8 inter-ISP connections on the overhead and peers slowdown. We consider the churn scenario described in section 3.3.2 and compute the overhead and peers slowdown for each experiments.

5.4.1 Impact on Overhead

We see in Fig. 10 that the churn does not increase the overhead, which means that with high locality values the overhead remains low even in presence of churn. Indeed, the overhead with 8 inter-ISP connections is dramatically lower than BitTorrent for all flashcrowd durations and for the two sets of peers. In particular, the overhead for the second set of peers (8 *connections, set* 2 in the figure), which corresponds to the peers arriving due to the churn, is lower than for the first set of peers (8 *connections, set* 1 in the figure). In fact, as peers from the first set stays 5 minutes as seed before leaving the torrent, they contribute to improve the piece diversity for peers in the second set. As we have discussed in section 4.1, a higher piece diversity results in a lower overhead.

In summary, with a high locality value, the overhead remains low even in presence of churn.

5.4.2 Impact on Slowdown

In Fig. 11 on the x-axis, the first 10 000 peers correspond to the set 1 and are ordered according to their slowdown. The second 10 000 peers correspond to the set 2 and they are also ordered according to their slowdown.

Fig. 11 shows that the curve that gives the slowdown for 8 inter-ISP connections never reaches 20 000. That means that either some peers have crashed or that some peers have never completed the download. A deeper inspection of our traces shows that for the experiments with a flashcrowd of 60s and 6 000s, 200 peers have crashed (that corresponds to two nodes that crashed, running each 100 clients). If we take apart those peers that crashed, we find that for 8 inter-ISP connections and a flashcrowd of 60 seconds, less than 1% of the peers from the second set never complete, with a flashcrowd of 600 seconds 8% of the peers never complete, and with a flashcrowd of 6 000 seconds 10% of the peers never complete. This is due to an issue in the BitTorrent protocol. As explained in section 2, a peer contacts the tracker for a new list of peers when its number of neighbors drops below 7 . Hence, if a clique of peers with at least 7 members gets disconnected from the rest of the overlay, the BitTorrent protocol will never connect back this clique to the rest of the torrent.

We observe in Fig. 11 that the smaller the flashcrowd phase, the lower the slowdown for the second set of peers. Indeed, when the first peers in the second set start, there is already a lot of peers from the first set still in the torrent, part of them acting as seed. Therefore, the capacity of service of the torrent when peers in the second set start is much larger than when the first peers in the first set start. Thus, a lower slowdown for the second set. However, when

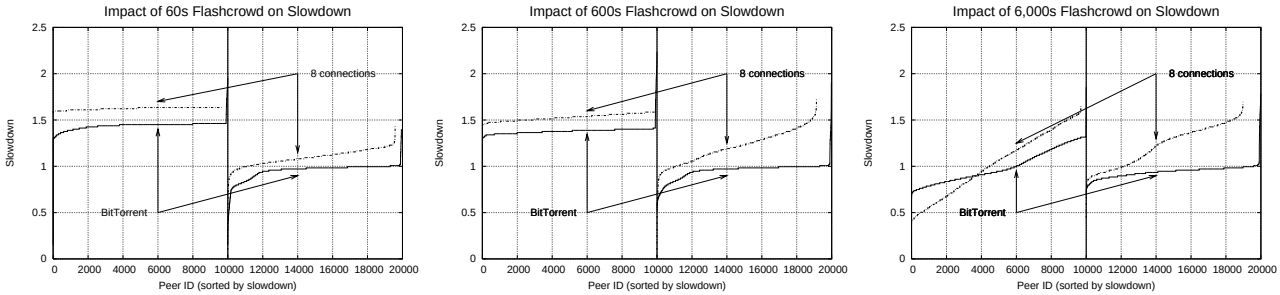

Figure 11: Slowdown with a torrent of 10 000 peers and 10 ISPs with churn. All peers in set 1 start with a uniform distribution within a specified flashcrowd duration of 60s (left plot), 600s (middle plot), and 6 000s (right plot). Each peer in set 2 starts when a peer from set 1 has completed its download. Set 1 and 2 consists of 10 000 peers and at most there are no more than 10 000 simultaneous peers in the torrent. We consider two scenarios: BitTorrent policy and locality policy with 8 inter-ISP connections. The lines represent the slowdown for each of the 20 000 peers joining the torrent as compared to the best achievable completion time. Peers are ordered according to their slowdown for each set (not according to their arrival time). *The churn prevent up to 10% of peers in the second set from completing.*

the flashcrowd duration is large (Fig. 11, right plot) peers in the second set start whereas not all peers in the first phase have started (we remind that the best download completion time is 83 minutes, thus 4 980 seconds). Therefore, as a fraction of peers in both sets share the torrent at the same time, they experience the same capacity of service from the torrent, thus an equivalent slowdown.

In summary, with a high locality value, a fraction of the peers may never complete their download in presence of churn. This is a major issue for any locality policies. However, we propose a solution to this issue in section 6.

5.5 Discussion

In this section, we have shown that the choice of high locality values is robust to the torrent size, the number of peers per ISP, and the presence of congestion on the inter-ISP links. Therefore, whereas high locality values allow a dramatic reduction in the inter-ISP traffic without any major impact on peers slowdown, they are viable in practice, because there is no need for an *a priori* knowledge of the torrent characteristics or of the network characteristics to determine the appropriate number of inter-ISP connections. According to our experiments a choice in the order of 80 inter-ISP connections is reasonable when the content provider wants huge inter-ISP traffic savings without any noticeable impact for the end-users. A choice in the order of 8 inter-ISP connections allows content provider to reduce the inter-ISP traffic to its minimum with a moderate impact on the end-users.

The impact of high locality values on the peers slowdown has to be understood as the one that will be achieved in the worst case. Indeed, in case of congestion on inter-ISP links, high locality values will lead to a better slowdown than the one achieved by the BitTorrent policy.

However, our experiments have shown that the churn may prevent a fraction of peers to complete with high locality values. We have identified that this is due to an issue in the BitTorrent algorithm that manages the reconnection to the tracker. This problem also exists with the BitTorrent policy, but is exacerbated by our locality policy. In section 6, we propose a solution to this problem.

6. BITTORRENT ALGORITHM EXTENSION

In this section, we discuss an extension to the BitTorrent algorithm that manages the reconnection to the tracker. This extension requires a few additional lines of code to the client and to the tracker. However, this extension is compatible with existing BitTorrent clients, so it can be incrementally deployed. In the following we describe the modification to the client, then to the tracker.

The client side algorithm that manages the reconnection to the tracker may end up in a deadlock. This deadlock, whereas unlikely to occur with the BitTorrent policy, is highly likely to occur with a locality policy. As mentioned in section 2, a peer recontacts the tracker every five minutes if it has less than 7 neighbors. According to our experiments presented in section 5.4, cliques of at least 7 peers appear in case of churn, which makes the peers belonging to those cliques never able to complete.

A simple way to solve this issue is to ask each client to recontact the tracker when none of its neighbor has downloaded a new piece for a given period of time T. It is easy for a peer to track the pieces that neighbors have downloaded thanks to the HAVE messages that are sent by a peer to each of its neighbor each time this peer has received a new piece. This client side algorithm modification allows peers to always detect partitions. In order to avoid an explosion of requests at the tracker [18] in case of partition, each peer that detects, after a period T, that none of its neighbor has downloaded a new piece will start a timer randomly chosen with a uniform distribution in the interval $[0, T]$. Then it will send a request to the tracker for new neighbors at the expiration of this timer. Each peer will perform an exponential backoff of T in case, after the request, it either do not receive an answer from the tracker or still do not see any new pieces among its neighbors. This way, even in case of a large torrent with missing pieces (due for instance to the failure of the initial seed), the tracker will not be overloaded

by requests from peers.

The choice of the initial value of T is a tradeoff between the time to react to a partition and the false detection of partitions due to a slow seed. An initial value of T set to 5 minutes sounds reasonable. Also, an upper bound of T set to a few hours is reasonable in case of backoff of T.

The modification to the client side algorithm is simple and does not require any specific support from the tracker. However, when the tracker implements a locality policy, it might take time for a clique in a given ISP to reconnect to the torrent, as the tracker will return to those peers local neighbors with a high probability. In order to prevent this issue, we propose also a simple modification to the tracker side algorithm when a client requests for a new list of neighbors in case of partition.

The tracker side algorithm modification is also simple. When a client contacts the tracker to receive a new list of neighbors due to a partition, the tracker will send back a random subset of neighbors without taking into account the locality policy. This way, it is likely that the clique will reconnect to the torrent. However, in order to prevent this mechanism to be abused by peers in order to bypass the locality policy (peers can lie when they say that they observed a partition), the tracker will return a random list of neighbors to a single peer per ISP every 10 minutes.

The detailed evaluation of this algorithm extension will be addressed in future work.

7. RELATED WORK

The load peer-to-peer traffic creates on ISPs has been discussed for a few years. However, the recent interest of content providers in peer-to-peer content replication has created a renewed high interest in the problem. There are mainly two categories of work in the field: evaluation and architectural works.

The evaluation works focus on understanding the impact of peer-to-peer traffic on ISPs and on how locality can help out of the context of a specific implementation of locality. Our work is part of this category. The architectural works propose new architectures to implement locality policies. Those works build on the results provided by the evaluation works. We present in the following related works belonging to both categories.

Evaluations

Bindal et al. [7] present the impact of a deterministic locality policy on ISPs' peering links load and on end-users experience. This work is the closest to ours, but differs in several fundamental points. In this work, the authors define a constant number of inter-ISP connections, which requires a peer-to-peer client modification in order, for the client, to guarantee that each ISP keeps its predefined number of inter-ISP connections. They show that, in the scenario they consider, their biased neighbor selection significantly reduces inter-ISP traffic with a minor impact on the end-users download time, as long as the seed is four times faster than a leecher. The authors considered simulations on a scenario with 14 ISPs with 50 peers each, thus a torrent of 700 peers. The main differences with our work is that we consider large scale experiments with up to 10 000 real BitTorrent clients. We go beyond the results presented in this work by evaluating a large range of locality values, the robustness of high locality to the torrent size, the number of peers per ISP, the presence on inter-ISP bottlenecks, and the presence of churn.

Karagiannis et al. [14] first introduced the notion of locality in the context of peer-to-peer content replication. First, they monitored the BitTorrent traffic flowing through the access link of an edge network. They show that $70 - 90\%$ of all the contents downloaded on the local network was downloaded from external peers. Then, they show that locality can actually improve performances for the end-users, as there is a high temporal overlap of users when they want to download a same content. Second, the authors used a log collected from a BitTorrent tracker [13] to simulate the traffic saving that could have been done with different content distribution systems. Their conclusion is that peer-assisted locality distribution is an efficient solution for both the ISPs and the end-users. Our work specifically focus on an extensive evaluation of a locality policy on real large scale torrents. We confirm that a locality policy significantly reduces the inter-ISP traffic, but we show that this reduction can be up to two orders of magnitude with a high locality value. In addition, we explore the impact of locality on the peers slowdown. In summary, our work provides new important insights on the impact of a locality policy for peer-to-peer content replication.

Finally, we show that an idealized model that assumes high piece diversity to simulate peer-to-peer content replication (as for instance the one considered by Karagiannis et al. [14]) may significantly underestimate the inter-ISP traffic overhead.

Architectures

P4P [21] is a project whose aim is to provide a light-weight infrastructure to allow cooperation between peer-to-peer applications and ISPs. The P4P architecture is based on two components: the ISP tracker (itracker) that is deployed by each ISP and the application tracker (app-tracker) that is deployed by the content provider. The itracker provides interfaces that allow peers and content providers to obtain information on the ISP's network. The app-tracker is responsible of the peer selection. It it similar to the BitTorrent tracker except that it selects peers considering both the application requirements and the itracker information. Results derived from P4P suggest that locality reduces peer-to-peer traffic by up to $50 - 70\%$ with increased performances.

Aggarwal et al. [5] present an architecture that is similar by some aspects to P4P. The authors define the notion of *oracle* that are supplied by ISPs in order to propose a list of neighbors to peers.

Another approach that requires no dedicated infrastructure is Ono [8]. Ono clusters users based on the assumption that clients redirected to a same CDN server are close. The authors have developed an Ono plugging for the Vuze client and they have reported up to 207% performance increase in average peer download completion time.

In all those approaches, only a fraction of the traffic is kept local in order preserve the robustness of the torrent. For instance, there are around 80% of intra-ISP connections for the P4P field tests. This moderate locality leads to inter-ISP

traffic savings around 60% for the P4P field tests. We show that we can push locality further, higher than 99%, in order to obtain significantly higher inter-ISP traffic savings, up to two orders of magnitude. We also monitor the entire torrent, which allows us to evaluate the inter-ISP traffic reduction at the torrent scale for each inter-ISP link. By having the total control on our torrents, we assessed the impact of the torrent size and of the number of peers per ISP on the overhead and slowdown. We confirmed that in case of inter-ISP links congestion, a locality policy can result in improved peers download completion time, but we show that when there is no congestion, high locality values lead to a peer slowdown close to the one achieve by the BitTorrent policy. Finally, we identified and proposed a solution to an issue in a BitTorrent algorithm that is responsible for the lack of robustness of BitTorrent with churn in case of high locality.

Our work is complimentary to all those previous architectural works as we do not propose a new architecture to implement a locality policy, but we show how far we can push locality and what will be the benefits of a high locality using such an architecture

Bickson et al. [6,10] introduce Julia, a locality aware peer-to-peer protocol that minimizes the latency across peers while the download progresses. Julia is slightly slower than BitTorrent and only reduces the inter-ISP traffic by 33%. HPTP [20] allows to cache peer-to-peer traffic using existing HTTP caches to decrease peer-to-peer overhead . However, it requires fundamental changes in the application layer protocol. Finally, György Dán [11] proposes a cooperative caching strategy to help ISPs reduce the load due to peer-to-peer traffic.

8. DISCUSSION

Our work is intended to be complimentary to previous works [8, 14, 21] by answering the fundamental question: How far can we push BitTorrent locality? In particular, we do not propose a new architecture or implementation of a locality policy, but we shed a new light on what happen if we push BitTorrent locality to the limit.

To perform the mapping between peers and their ISPs, we rely on existing propositions as for instance P4P [21], the technique used by Ono relying on a CDN infrastructure [8], or ALTO [1] an on-going work of the IETF. Even if our evaluation relied on a tracker modification, our results are not limited to this implementation. In particular, all our results still hold if the locality policy is implemented by the clients. However, we argue that even if in today's context, it is faster for modifications to be deployed in clients than in trackers, it makes more sense for a content provider to enforce locality without requiring a modification of each and every client versions connecting to its torrent. Indeed, the content provider has control on the tracker and the initial seed, but not on the BitTorrent clients used by the peers, unless the content provider also enforces the use of its own proprietary client.

In this paper, we have performed an extensive evaluation of the impact of high locality values on inter-ISP traffic and peers download completion time. We have run experiments with up to 10 000 real BitTorrent clients in a variety of scenarios.

Our main findings are the following.

- The overhead decreases linearly with locality. The saving on the inter-ISP traffic is up to two orders of magnitude for high locality values compared to the BitTorrent policy. Moreover, this significant saving has no negative impact on the peers slowdown.
- The capacity of the initial seed is critical to have a low peer download completion time and overhead with a high locality.
- Models that assume good piece diversity might significantly underestimate the overhead.
- High locality values enable a low overhead and slowdown in a large variety of scenarios. In particular, they are robust to the torrent size, the number of peers per ISPs, and the presence of congestion on inter-ISP links.
- In the worst case, high locality values lead to a higher peers slowdown than the one achieved by the BitTorrent policy, but in case of congestion, high locality values enable a lower slowdown than the one achieved by the BitTorrent policy.
- In case of churn, a fraction of peers do not complete with high locality values. However, we identified the issue and proposed a solution that consist in a modification of the BitTorrent algorithm that manages the reconnection to the tracker.

Our results provide new insights to content providers and ISPs. Content providers have the control of the tracker and the initial seed. As a consequence, they have to sufficiently provision the initial seed in order to reap full benefits from the locality policy implemented at their tracker. The ISPs will benefit significantly from high locality even for torrents with a few number of peers per ISP. For large torrents, ISPs will experience a dramatic saving on inter-ISP links. Finally, content providers and ISPs can easily promote locality to end-users, as they will experience an equivalent or even shorter download completion time when the network is congested.

Acknowledgments

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see https:// www.grid5000.fr)

9. REFERENCES

[1] Application-layer traffic optimization (alto). http://www.ietf.org/html.charters/alto-charter.html.
[2] Bittorrent, inc. http://www.bittorrent.com.
[3] Grid'5000. https://www.grid5000.fr.
[4] Instrumented bittorrent client. http://www-sop.inria.fr/planete/Arnaud.Legout/Projects/p2p_cd.html.
[5] V. Aggarwal, A. Feldmann, and C. Scheideler. Can isps and p2p users cooperate for improved performance? *Proc. of CCR*, July 2007.

[6] D. Bickson and D. Malkhi. The julia content distribution network. In *Proc. of WORLDS'05*, San Francisco, CA, USA, December 2005.

[7] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proc. of ICDCS'06*, Lisboa, Portugal, July 2006.

[8] D. R. Choffnes and F. E. Bustamante. Taming the torrent: A practical approach to reducing cross-isp traffic in p2p systems. In *Proc. of ACM SIGCOMM*, Seattle, WA, USA, August 2008.

[9] B. Cohen. Incentives to build robustness in Bittorrent. In *Proc. of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, June 2003. http://bitconjurer.org/BitTorrent.

[10] D. M. D. Bickson and D. Rabinowitz. Locality-aware content distribution. Technical Report TR 2004-52, School of Computer Science and Engineering, The Hebrew University, 2004.

[11] G. Dan. Cooperative caching and relaying strategies for peer-to-peer content delivery. In *Proc. of IPTPS*, St. Petersbourg, FL, USA, February 2008.

[12] M. Dischinger, A. Mislove, A. Haeberlen, and P. K. Gummadi. Detecting bittorrent blocking. In *Proc. of ACM IMC*, Vouliagmeni, Greece, October 2008.

[13] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM'04*.

[14] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *Proc. of IMC'05*, Berkeley, CA, USA, October 2005.

[15] A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and Sharing Incentives in BitTorrent Systems. In *SIGMETRICS'07*, San Diego, CA, USA, June 2007.

[16] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest First and Choke Algorithms Are Enough. In *IMC'06*, Rio de Janeiro, Brazil, October 2006.

[17] A. Odlyzko. Internet pricing and the history of communications. *Computer Networks*, 36:493–517, 2000.

[18] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proc. of IPTPS*, Ithaca, NY, USA, February 2005.

[19] P. Rodriguez, A. Kirpal, and E. W. Biersack. Parallel-access for mirror sites in the internet. In *Proc. of INFOCOM*, Tel-Aviv, Israel, March 2000.

[20] G. Shen, Y. Wang, Y. Xiong, B. Zhao, and Z.-L. Zhang. Hptp: Relieving the tension between isps and p2p. In *Proc. of IPTPS*, Bellevue, WA, USA, February 2007.

[21] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4p: Provider portal for applications. In *Proc. of ACM SIGCOMM*, Seattle, WA, USA, August 2008.