Delft University of Technology
Parallel and Distributed Systems Report Series

It's not that simple: an empirical study on the influence of locality on download speed in BitTorrent

Maciej Wojciechowski, Johan Pouwelse
j.a.pouwelse@tudelft.nl

Completed September 2009.

Report number PDS-2011-008

ISSN 1387-2109

Published and produced by:
Parallel and Distributed Systems Group
Department of Software and Computer Technology
Faculty Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

Information about Parallel and Distributed Systems Report Series:
reports@pds.ewi.tudelft.nl

Information about Parallel and Distributed Systems Section:
http://www.pds.ewi.tudelft.nl/

Abstract

Traffic localization is currently considered a promising approach to decrease the load of P2P systems on the Internet backbone. A number of solutions that promote P2P-locality have therefore been proposed. However, their evaluation has been largely focused on simulations with limited validation against real-world systems.

In this paper we empirically measure the influence of locality on end-user download speed in BitTorrent. For that, we design Proximity Toolbox, a set of tools that uses cached round-trip-time measurements to estimate latency and feed close peers to a regular BitTorrent client. We run experiments from five vantage points in North America, Europe and Asia downloading real content using real-world connectivity.

We find that locality influences end-user speed in different ways in different locations: it has no effect in one location, generates significant increase (up to 57.65% for the average) in three out of five, but also leads to speed decrease (up to 37%) in one location. Furthermore, we show that this diversity of effects is masked if one overlooks the diversity of vantage points: analyzing all locations together, one can observe a significant positive influence of locality on download speed.

Our measurements reveal a more complex picture of P2P-locality than the one considered in the design and evaluation of previous locality-based solutions. Even with few measurement points, we are able to contradict the claim that P2P-locality does not lower end-user performance. This, in turn, points to the need for further research on the factors determining the effectiveness of locality.

Contents

List of Figures

List of Tables

1 Introduction

P2P applications, especially BitTorrent [1], generate a large fraction of the Internet backbone load [2]. Long distance Internet traffic often has to traverse many Autonomous Systems, generating cost. What is more, it is widely believed that BitTorrent creates congestion on backbone links and therefore deteriorates performance of the whole network.

Current implementations of the BitTorrent protocol do not take physical network information into account when choosing peers to exchange data with. This means that data may be often transfered between continents while it may be available in the user's close proximity, e.g., the same Internet Service Provider (ISP) or city. This is suboptimal from the cost perspective. Many ISPs and researchers have argued that if P2P traffic could be kept more local, backbone load would decrease. It is widely believed it would lead to lower cost of operating the network and also enhance performance for end-users.

P2P-locality refers to traffic localization in peer-to-peer networks. The basic concept is to change peer selection so that more data is exchanged with peers that are local. Local has different meanings: it can be geographical closeness, number of IP hops, number of traversed ASes, cost of transporting the bytes to the other end or latency of the connection.

There are some developed and/or deployed P2P-locality solutions. They range from positioning on the Internet overlay [3], through changing the tracker behavior [4] to ISP-aided solutions [5] [6] [7]. One issue with all these solutions is that they do require either large deployment or cooperation of a third-party in order to be tested. Because of that, designers often assess the performance of their systems using simulations.

In this paper we measure the influence of locality on download speed from end-user perspective. We run real-world experiments with real content, real peers and real connectivity. Our goal is to study how P2P-locality can affect particular users in the current Internet. We find that the way traffic localization influences download speed is more complex than previously reported. Our findings show the need to revisit the existing simulations in order to better capture the diversity of the Internet network and BitTorrent protocol.

We study locality in five different vantage points: Canada (CA), Hong Kong (HK), the Netherlands (NL), Poland (PL) and Romania (RO). Depending on the location, locality has a different effect on the end-user performance. There is significant performance increase in Canada, Poland and Romania (up to 57.65% for the mean). However, there is only a minimal speed change in Hong Kong and a significant performance decrease in the Netherlands (up to 37%). In most cases, locality, or lack thereof, is more important than the particular level of locality. The exception in our vantage points is Canada where only higher locality levels lead to performance change.

In addition to studying locality in five locations separately, we contrast this analysis with one that averages all vantage points without discriminating them. Surprisingly, the diversity of how locality influences the download speed is not reflected in the global perspective—all levels of locality lead to statistically significant performance increase. This demonstrates how one can oversee important details by looking only at the global picture.

In order to do the measurements, we design Proximity Toolbox: a set of tools for finding local peers and feeding them to a regular BitTorrent client. Proximity Toolbox uses cached round-trip-time (RTT) measurements to estimate locality and later, during the download, orders the peers according to their latency, giving preference to peers considered to be closer. We use traceroute to measure the latency and a regular BitTorrent client to download the content.

Our first contribution in this paper is a measurement study showing a more diverse nature of P2P-locality than presented in previous studies. The second contribution, Proximity Toolbox, is a means to achieve and measure locality in real-world conditions.

The remainder of the paper is structured as follows. In Section 2 we study the existing research in P2P-locality and show how our findings extend current knowledge in the field. We describe peer discovery and peer selection in BitTorrent in Section 3. In Section 4 we show how we use latency for locality and in Section 5 we show how we estimate locality in Proximity Toolbox. Then, in Section 6 we describe the setup we used to do P2P-locality measurements: infrastructure, client settings, swarm selection algorithm, etc. We analyze the results of the measurements, using statistical model, in Section 7. Finally, we conclude in Section 8.

2 Related work

Two main approaches for achieving P2P-locality have been considered in the literature. The first is to bias peer-selection towards peers with low latency. This concept was used by Choffnes et al. to implement Ono [3]. Ono uses RTT measurements to DNS servers of a Content Delivery Network to establish the virtual coordinates of a node. Using the coordinates, Ono estimates the distance between peers and gives priority to peers that are considered to be *close*. Choffnes et al. implemented the system as a plugin to the Azureus BitTorrent client and had this plugin installed by more than 120 000 clients, making it the largest P2P-locality trial conducted so far. Peers selected by Ono had significantly lower latency than random peers (median latency over two orders of magnitude lower). However, clients using Ono did not observe better average download time. In fact, the download performance slightly dropped. One notable exception was a Romanian ISP that had significantly higher bandwidth inside the ISP than to the outside world—an obvious case for P2P-locality deployment.

A variation of the same approach is constraining BitTorrent peers to select mostly peers from the same ISP. Bindal et al. [8] used this approach to limit the number of connections a peer can have to peers from outside the same ISP. In order to assess the performance of their system, they ran simulations on a fully connected graph of 14 ISPs, with 700 hundred peers distributed between them. They concentrate mostly on decreasing data redundancy inside the ISP, but in some cases they notice around 30% average speedup. The simplifying assumptions of this work leave as an open question what is the effect of ISPs heterogeneity and topology. In this paper we argue that because P2P users tend to be concentrated in some parts of the world, there is very limited potential for P2P-locality in some locations.

The second approach to achieve P2P-locality is to use ISP-provided topology information. P4P [6] proposes such a cooperation between P2P users and their ISPs: ISPs would provide servers that their users would be able to query. The ISP would tell the P2P users which peers are close. A similar approach was proposed by Aggarwal et al. [5]. Unsurprisingly, the P4P-like approach has got a lot of support from Internet Service Providers: they have a chance to decrease the cost of operating their networks with very little change to the infrastructure. IETF ALTO working group [7] is currently trying to develop a topology-related information exchange protocol for communication between P2P users and the ISPs. In the "Mythbustering Peer-to-peer Traffic Localization" Internet Draft [9], it is considered "very likely" that P2P-locality will increase the application performance, especially for large swarms and networks with large capacity. In this paper we argue that while this claim is mostly true, the situation is not that simple.

P4P approach has been used in the Comcast-Pando trial [10], a second (next to Ono) locality deployment in the real-world. Pando is a commercial P2P software for file distribution [11]. In the trial, clients downloaded a 20MB file. A modified tracker was used to provide locality—instead of returning random peers, the tracker used topology information (provided by Comcast) to return local peers. The trial has shown significant speed improvement: up to 15% globally and up to 85% for Comcast users. Even though the file was very small and many details of the trial

remain unknown, it is a strong evidence that locality can boost the download speed, especially for ISPs with many P2P users.

We observe that in the aforementioned studies, irrespective of how locality is achieved, evaluation predominantly or solely focus on the effects of locality on ISP costs. It is often a premise that an improvement on the end-user's service will follow IPS's cost reduction. In contrast with this picture, we understand that the assumption that P2P-locality is always a win-win situation for ISPs and end-users needs further evidence. In particular, none of the discussed research has considered the possibility that locality influence on download speed varies depending on network conditions.

This study addresses this gap, concentrating on how P2P-locality affects end-user performance in distinct vantage points on the Internet. By individually analyzing results from different experiment locations, we show that the influence of locality varies.

3 Bittorrent

In this section we explain BitTorrent and the peer exchange (PEX) extension. We concentrate on peer discovery and peer selection as those are the only aspects of the protocol we are concerned with.

BitTorrent is currently the most popular P2P file sharing protocol. The protocol is highly decentralized: there is no global bootstrap servers or repositories. Instead, users downloading the same content constitute a swarm—a separated group of peers exchanging pieces of the same file. Two different peers, not being part of the same swarm, do not communicate with each other and two peers in the same swarm exchange only bytes belonging to the given swarm.

In order to join a swarm the user needs to obtain a .torrent file—a metadata catalog containing i) information about the content being shared, ii) address of one or more trackers, and iii) checksums necessary to verify the downloaded file pieces. The BitTorrent protocol does not define a method for distributing the metadata files; normally .torrent files are distributed through websites. In order to join a swarm, the user needs to connect to the tracker found in the metadata file. The tracker is a server, whose only responsibility is to keep track of users in the swarms. It does not host the content, it does not download it and it does not need to know what content is being tracked. When contacted by a user, the tracker responds with a list of IP addresses of other peers in the swarm. The user then connects to the peers received from the tracker, establishes BitTorrent sessions with them and can start downloading the file. Normally, both the tracker and the client choose IP addresses randomly—the tracker returns random peers from the swarm and the client connects to them in random order. The list of IP addresses sent by the tracker is usually between 50 and 200 peers.

In this paper, we only modify the peer discovery and the peer selection parts of BitTorrent. No changes are made to any other parts of the protocol. Hence, we do not describe the rest of the BitTorrent protocol.

The tracker is the standard way of discovering peers in BitTorrent. In addition, all major clients also support the peer exchange (PEX) extension: a gossip-based protocol for intra-swarm communication, allowing clients to discover other peers in the same BitTorrent swarm. PEX is a simple extension that allows fast peer discovery without the support of the tracker. Two peers supporting PEX exchange their sets of connected peers; once immediately after connecting and incrementally later on. Note that only connected peers are exchanged—this prevents spreading addresses of peers that are no longer active in the swarm.

Using PEX one can implement a PEX-crawler—a client that tries to find all the peers in the swarm. It can be done by connecting to more and more peers and using PEX messages to

extend the known peers set. Indeed, our experiments show that after five minutes our PEX-crawler discovers more than 90% of the number of peers reported by the tracker. In practice, PEX-crawler is a useful tool for creating a snapshot of peers in the swarm at a given moment.

The tracker list of 50-200 peers is insufficient for a client-side locality solution. A bigger list of peers is necessary in order to find peers that are local. One possible way to obtain addresses of more peers in the swarm is to query the tracker many times. We believe that using the PEX-crawler is both a more robust and a more reliable method. It also does not generate unnecessary load for the tracker.

PEX allows us to get a list of most of the peers in the swarm. Instead of having between 50 and 200 peers, we can have thousands of them.

4 Using latency for locality

In this section we describe the algorithm Proximity Toolbox uses to establish a locality-based order of peers in the swarm. The algorithm is based on traceroute measurements and Border Gateway Protocol (BGP) prefixes, both of which are explained. A naïve approach to ordering, based on pings, is not satisfactory, as we will show.

We use round trip time measurements as it is the easiest way of achieving locality. It does not require cooperation of other peers or use of a third party service. Choffnes et al. [3] have shown that using a latency-based approach does in fact decrease the number of IP hops, proving the validity of the approach.

4.1 Straw man solution

Let us first consider a naïve approach. When a client gets a list of peers in the swarm, it starts to send ICMP echo packets (pings) to all of them. Ping is a very simple tool for measuring the RTT between two hosts. Peers can be then ordered according to the measured RTT.

Such a naïve approach seems to do the job: it orders the peers according to the latency, achieving some level of locality. Unfortunately, there are several reasons why it would have only limited success:

- Pings are not accurate: they are highly affected by congestion, especially by the congestion of the end-user access link.
- Many hosts do not respond to pings: our measurements show that only around 40% of peers actually responds to ICMP echo packets. This means that only around 40% of peers can be ordered and the RTT value for the rest of them would be unknown.
- BitTorrent tends to have very high churn: by the time the pinging is done, many peers in the swarm have already changed. This makes the RTT list outdated and the whole process needs to be repeated.

The high churn problem could be overcome with caching: instead of measuring peers every time they are encountered, one could store the RTT for future use. Unfortunately, our tests show that the rendez-vous probability in BitTorrent is very low: two swarms with sizes above 20 000 peers usually do not have more than a few common peers. Taking into account that BitTorrent is used by millions of users, such a cache would very fast grow to unmanageable sizes.

4.2 Addressing the issues of the straw man solution

In this subsection we address the three problems with naïve solution: inaccuracy, low response rate and high churn in BitTorrent.

Traceroute

The main reason why the naïve approach is inaccurate is last-mile congestion. Last-mile is a common name for the link between the end-user and its provider. Very often the last-mile is a an ADSL or Cable link, both of which have larger router queue sizes and different packet drop policies than backbone links. A larger queue size means that packets are delayed more before being sent out, increasing the latency of the end-to-end link. Last-mile links also have very limited capacity. This, together with high utilization, typical for BitTorrent users, results in high congestion of the last-mile links.

Simple end-to-end pings are not the only way of measuring latency. One example of a more sophisticated use of ICMP echo packets is traceroute—a net diagnostic tool capable of inferring the full router path between two end-hosts. Traceroute modifies the time-to-live (TTL) value of packets, causing the routers on the way to drop them and send an ICMP time exceeded packet back to the sender. Using different TTL values, traceroute can infer all the routers on the path between two end-hosts.

Using traceroute it is possible to get the address of the *last-router* before the end-point. This is the ISP-controlled side of last-mile link, not suffering from last-mile congestion problem.

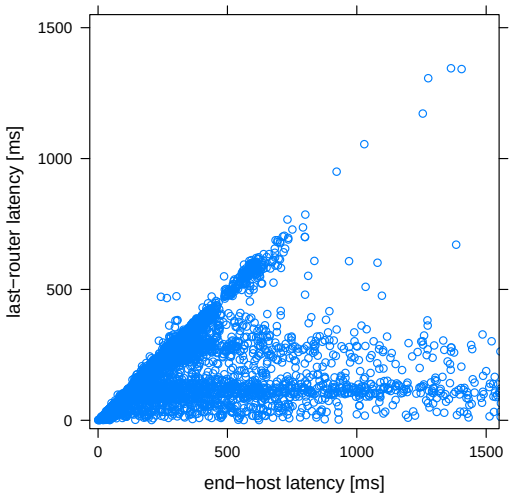


Figure 1: *Comparison between* last-router *and end-host measurements for 10 000 end-hosts. 2.5% of end-host measurements were higher than 1500 ms and are omitted from the graph.*

In order to illustrate the last-mile congestion problem, we conducted a small experiment on 10 000 P2P users. Figure 1 compares RTT measured to the *last-router* and to the end-user, for the same IP address. In 48% of cases the difference between end-host and *last-router* measurement is more than 25 ms—in congestion-free conditions, 25 ms corresponds to roughly 1000km in geographical distance. We believe this discrepancy is caused by the last-mile congestion, which contributes disproportionally to the end-to-end RTT measurements.

It has to be noted, however, that traceroute cannot successfully identify the *last-router* if the end-host is not responding to ICMP echo packets. In that case traceroute can infer only some hops on the path; often all hops but the last one: the end-host. Unfortunately, it is hard to distinguish such a situation from one where one router on the way is blocking the traceroute. Because we cannot be sure that we in fact find the *last-router*, we discard all the measurements where the end-point is not responding.

BGP prefixes

The second component of our latency-estimation algorithm is based on Border Gateway Protocol (BGP) prefixes.

BGP is the *de facto* standard routing protocol of the Internet. It is used to route packets from one destination to another. Different organizations, called Autonomous Systems (ASes), have control over different networks. They exchange information about the networks they control and how to reach them. In the BGP terminology networks are usually called *BGP prefixes*. A prefix denotes one network and for every IP address it is clear whether it is a part of a given prefix. When a packet is sent, first it is delivered to the AS in control of its destination prefix and then the AS delivers the packet to the right end-host.

The list of prefixes can be easily obtained. Proximity Toolbox uses the prefix list published by CAIDA[12].

In this paper we assume that two hosts in the same network (prefix) are close to each other and therefore latency to them should be almost identical. For example, if two hosts 10.0.1.1 and 10.0.1.2 are in the same prefix 10.0.0.0/16 and measured RTT to 10.0.1.1 is 53 ms, we assume that 53 ms is a good estimate of latency to 10.0.1.2 as well. BGP does not prohibit a prefix to be spread over large geographical area, but according to our experience this is not often the case for end-user prefixes. Later in the paper we show that our BGP prefix based solution does, in fact, select peers that are geographically close. Note that in the above example 10.0.0.0/16 is the most specific BGP-prefix and 53 ms is *last-router* RTT, free from last-mile congestion.

Thanks to BGP prefixes there is no need to know the exact RTT to every peer in the swarm. It solves the problem of low response rate to ICMP echo packets. We find that even though we cannot directly measure the RTT to 60% of peers, the reachable peers ensure coverage for the overwhelming majority of BitTorrent-active BGP prefixes.

Caching

The third problem of the straw man approach—high churn—can be overcome with caching in combination with BGP prefixes. We build a database of RTT measurements to many BGP prefixes. To do that we extract IP addresses from BitTorrent swarms, using PEX-crawl. For every of those IP addresses we measure RTT to the end-host and *last-router* by sending traceroutes. The number of traceroutes needs to be sufficient to avoid short-time congestion while not generating too much overhead. We choose 40 as a reasonable trade-off between the two and we store the best result in the database. Sometimes, because of how routers are handling ICMP packets, the end-host measurement is better than the *last-router* measurement. Hence, we always choose to store the better of the two in the database.

Data in the database is stored per prefix, not per IP address. That means that two measurements, for two different IP addresses from the same prefix, refer to the same record in the database. There are theoretically roughly 2^{32} IP addresses, but only around 300 000 BGP prefixes. Because the RTT is stored on prefix level only, the database size is kept considerably small.

Location	IPs	prefixes	ASes	hit ratio
CA	430 862	35 428	4844	0.93
HK	251 732	29 327	4400	0.89
NL	403 047	36 245	5307	0.91
PL	362 173	33 786	4856	0.90
RO	222 769	24 195	3950	0.82

Table 1: Number of unique IP addresses measured per location, together with number of unique prefixes, ASes and hit ratio.

Whenever an RTT value for an IP address is needed during the download phase, Proximity Toolbox looks for cached RTT value for appropriate prefix and uses that value as latency prediction for the given IP address.

5 Estimating locality with the Proximity Toolbox

In this section we describe the latency measurements we have done using Proximity Toolbox. We show the coverage of IP addresses and BGP prefixes we achieve. We also study how our latency database can estimate locality and how the observed RTT values differ between locations.

5.1 Coverage

In order to build a latency database, we run our latency estimation algorithm for a few days at each location. Table 1 shows the number of unique (responding) IP addresses measured, together with the number of prefixes and ASes covered by those IP addresses. For every location we also calculate the level of swarm coverage: the hit ratio. The hit ratio is a quality measure of the database. It tells us for what fraction of IP addresses, from a random sample of peers, the database has a latency prediction.

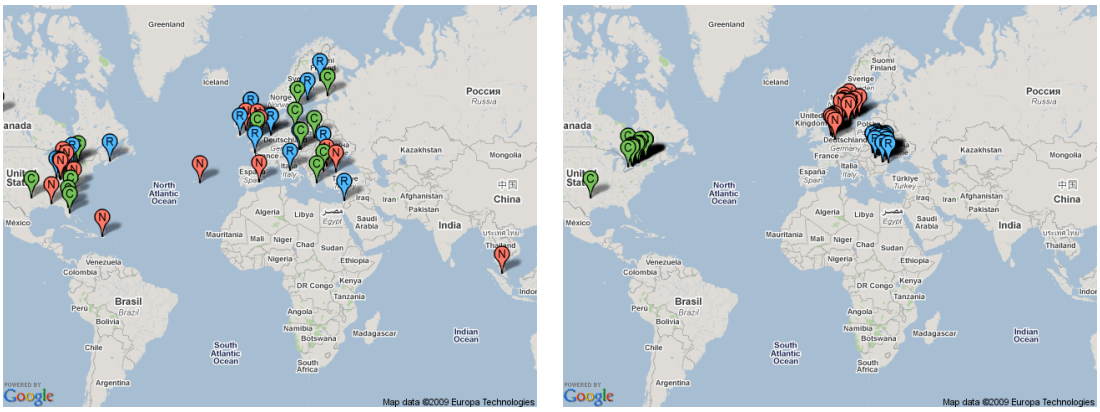
There is a significant discrepancy between the hit ratio in Romania and other locations: because of a technical problem less IP addresses have been measured there. However, as we show later in the paper, the difference in hit ratio is not correlated with the effect locality has in particular locations. Indeed, Romania, having lowest hit ratio (0.82), has the highest speed increase, while the Netherlands, having second highest hit ratio (0.91), suffers from significant performance decrease.

5.2 Validation

Our latency-estimation solution is based on a premise that two end-hosts, with IP addresses in the same prefix, are close to each other in physical and topological distance. Unfortunately, it is impossible to test this premise for the peers we measure.

In order to gain more confidence in our algorithm, we cross-check the locality predictions with an independent database. We use MaxMind GeoIP [13] to match IP addresses of peers to geographic coordinates and observe geo-dispersion of peers. We use geographical proximity as a proxy for topological proximity.

We run our locality prediction algorithm, for a 5000 peers swarm, in three different locations: Canada, the Netherlands and Romania. Figure 2 shows that, indeed, peers selected by Proximity Toolbox are very close to the measurement point.



(a) Random peer selection (b) Localized peer selection

Figure 2: Comparison of random and local peers selected from the same 5000 peers swarm by machines in Canada (C), the Netherlands (N) and Romania (R) (a) shows random peer selection with most of the peers in North America and Europe and some in south east Asia (2 peers in the far east and 4 peers on the US West Coast could not be fitted on the map). (b) shows local peers, all close to the appropriate vantage point. In fact, all peers selected by the Romanian vantage point are located in Romania.

The fact that the algorithm chooses close peers gives us confidence that our approach, based on traceroutes and BGP prefixes, is correct.

5.3 Latency distribution in various locations

Measured latency values differ significantly between locations. Figure 3 shows the cumulative distribution function (CDF) of latencies for all identified BGP prefixes in five vantage points. In Hong Kong we observe much higher RTT values (mean observed RTT is 311 ms) than in other locations (between 101 and 135 ms).

We attribute this difference to the fact that most peers we see (and therefore BGP prefixes we measure) are located in Europe and North America. Hong Kong is far from both and therefore the RTT values to many prefixes are higher.

We expect to observe similarly high RTT values in other regions that are far away from Europe and North America. The fact that P2P users are not uniformly distributed around the world, but rather concentrated on two continents, can seriously reduce the effectiveness of locality solutions: there is not much space for locality if there are almost no local peers. We believe this is an issue that must be considered in the design and evaluation of P2P-locality solutions.

6 Experimental setup

In this section we explain how we download the content (with several locality parameters), how we choose the swarms and what settings we use for the BitTorrent client. The clients use latency databases (created using the process described in the previous section) to select close peers in the swarm.

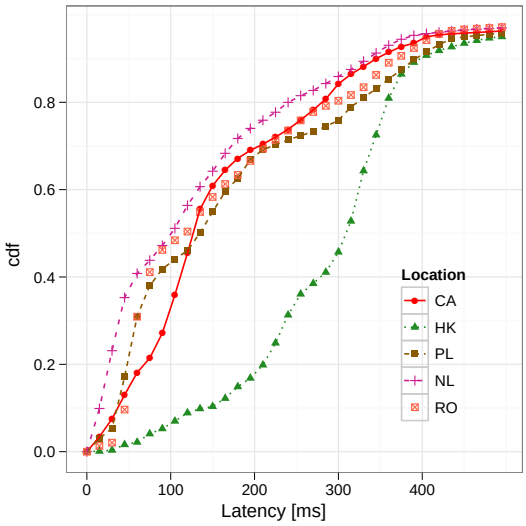


Figure 3: *Cumulative distribution of latencies in all five vantage points. Measured latency to one prefix is a single sample. There is a significant difference between latencies observed in Hong Kong and in other locations.*

6.1 Single experiment description

We want to measure the performance from an end-user perspective. We do that by running a series of downloads of various pieces of content. For a single machine and a given swarm, a single experiment consists of two steps:

1. harvesting IP addresses of peers in the swarm using the PEX-crawler
2. downloading the content using the libtorrent/rtorrent BitTorrent client[14]

Before joining the swarm, Proximity Toolbox gathers a list of peers using a PEX-crawler. The crawler runs for 5 minutes, getting on average more than 90% of peers in the swarm. Next, the list of addresses created by the crawler is randomly divided into 4 groups and a BitTorrent client is started for every one of the four groups. Each of the 4 clients starts with a different locality level: 0, 33, 67 or 100%. 0% is the typical BitTorrent behavior: peers are selected at random. 33%-locality means that every third peer will be chosen to be as close (latency-wise) as possible, while two thirds will be random. There will be two thirds of local peers for 67%-locality and only the most local peers for 100%-locality. All four instances are running on the same machine, at the same time, without any communication between them.

We do not want four different instances to compete for the same peers and therefore every client uses different parts of the swarm for downloading. Note that even though we split the swarm into 4 parts, every client still has a large list of peers available: between few hundreds and few thousands, depending on the swarm (we discuss the swarm sizes later in this section).

We stop the downloads after 30 minutes. In Section 7.2 we show that 30 minutes is a reasonable time to find out how locality settings influence the download speed.

6.2 Client settings and infrastructure

In the download phase of our experiment, we use a modified version of the libtorrent/rtorrent [14] BitTorrent client. We extended the software adding verbose logging capabilities and a latency-aware peer-selection algorithm. We did not make any other changes to the client. Peer selection is implemented in Python and runs as a part of the libtorrent library.

The client does not contact the tracker during the downloads. Instead, it receives the IP addresses from the PEX-crawler.

We use 50 kB/s upload and 400 kB/s download speed limits. We believe it is similar to an ordinary end-user connection in Europe and North America. Note that upload speed has indirect influence on achieved download speeds, due to the tit-for-tat strategy implemented in the BitTorrent protocol.

A normal BitTorrent client can have both incoming and outgoing connections. Our client does not accept incoming connections: they would introduce noise in the measurements, decreasing the effect different locality levels have on the results. Moreover, 50% of peers are not able to accept incoming connections, as a results of widespread NAT and firewall deployment in the current Internet [15]. Hence, we think that our decision to have only outgoing connections is justified.

We use five vantage points for our measurement: Canada, Hong Kong, the Netherlands, Poland and Romania. In all locations we have at least 100 Mbit/s connectivity, which is sufficient to concurrently run 4 clients with 400 kB/s download speed. Hardware resources usage during the experiments is minimal.

6.3 Swarm selection procedure

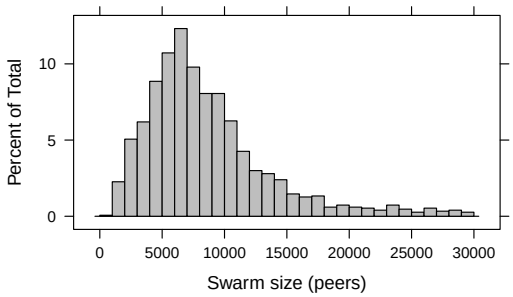


Figure 4: *Distribution of swarm sizes in the experiment. Swarms between 3000 and 13 000 peers constitute 77% of all swarms.*

Each machine in each vantage point must choose the swarms to join. Client-side locality is not possible for small swarms: if the list of peers is short, the client connects to most of them and prioritization is irrelevant. Moreover, in smaller swarms there is lower probability of finding peers that are really close or even in the same ISP of the vantage point. Hence, Proximity Toolbox uses only big swarms for the measurements. Figure 4 shows the distribution of sizes of the swarms in our experiment, with the mean size being 8757 and median size 7470 peers. The fact that Proximity Toolbox splits the swarm in four random parts makes the usage of biggest available swarms even more important. Cuevas et al. [16], found that almost 90% of all measured

swarms in the Mininova [17] community have less than 50 peers—there is no room for locality in swarms that small. They also observed that the 1000 largest swarms hold collectively around 40% of all peers, while constituting only 1% of all swarms. Using the biggest swarms allows us to contact more peers and better explore locality.

During the experiment, on every server, Proximity Toolbox chooses from the list of largest swarms available at a given time. In order to avoid constantly meeting the same peers, the same torrent is never downloaded more often than once every 15 hours. Our tests show that in two consecutive downloads around 90% of the peers are different.

All vantage points where we run Proximity Toolbox choose from the same set of available torrents. We do not explicitly prohibit two instances from downloading the same piece of content at the same time, but we add a nondeterministic factor to the swarm selection algorithm, to prevent all the instances from joining the same swarm simultaneously.

In total, the measurements resulted in 172 unique pieces of content. The most popular file was downloaded 53 times while 59 pieces of content were downloaded only once. We downloaded content in 6260 swarms (1252 per location). In total, for all locations, we downloaded 1675 gigabytes and uploaded 316 gigabytes of data.

7 Results analysis

In this section we use statistical analysis to evaluate how locality influences the download speed in our experiment.

7.1 Achieved locality level

Before we analyze the download speed, we first perform a sanity check to test to what extend Proximity Toolbox localizes the traffic. We use a *kilometers per packet* metric to compare the number of kilometers packets in a particular download had to "travel" over the network. We use physical distance between two end-hosts (calculated using GeoIP database) as a lower-bound on network-distance between them. Figure 5 presents the average number of kilometers per packet for all levels of locality in the five locations.

In all locations, besides Hong Kong, we observe an almost linear drop in average kilometers per packet. Values for 100%-locality are around one fourth of those for 0%-locality (which corresponds to random selection, currently employed in BitTorrent). This decrease is evidence that our latency-based approach does achieve a very good locality degree.

In Hong Kong we did not manage to achieve better kilometer per packet values. Indeed, for random peer selection (0%-locality) the average number of kilometers per packet is 3 times higher in Hong Kong (13281km) than in the Netherlands (4664km). The number of kilometers goes even higher for 100%-locality level (14737km). A simple explanation is that there are not enough hosts that are local to our vantage point in Hong Kong and it is not possible to exploit locality. It is also possible that this particular machine has poor connectivity to local peers while having good connection to North America or Europe. The fact the we were unable to achieve locality in this location is very significant: it is further evidence that a solution that works very well in four locations does not necessarily yield satisfying results in the fifth.

7.2 Warm-up and steady state

At the beginning of each download, the BitTorrent client looks for peers to exchange bytes with. Many peers need to be contacted: some do not respond at all and others have a very limited upload capacity. It takes time before the client converges to a more or less steady state.

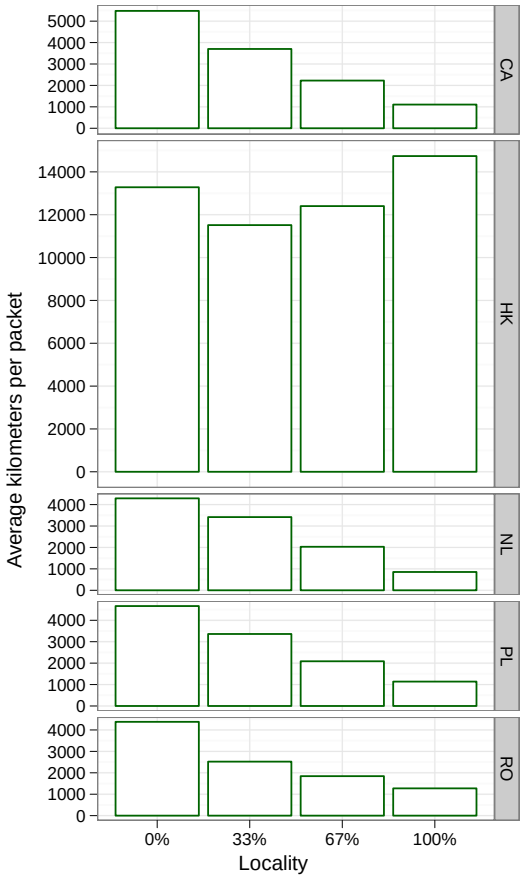


Figure 5: Average number of kilometers per packet according to GeoIP. We observe linear correlation with the level of locality for all locations apart from Hong Kong. In Hong Kong we observe both much higher number of kilometers for random selection and no decrease when using locality.

Note that in our measurements we stop every download after 30 minutes. Startup time can have different contributions in different swarms and we do not want the startup time to affect the measured speed. Our tests have shown that average speed in the first 5 to 10 minutes of the download is much lower than in any interval after the 10th minute. To be on the safe side, we discard the first 15 minutes of the download and use the second 15 minutes to calculate the average speed.

There are a few downloads which finish quicker than 30 minutes. We cannot discard them, because they represent an important group of very fast downloads (close to maximum speed) with short startup times. Instead of using the last 15 minutes, we use the second half of the download for them.

Throughout the rest of the paper whenever we refer to download speed, we mean the average speed in the second part of the download.

7.3 Performance metric

A typical way to study speed in BitTorrent is to compare average download speeds. However, according to our findings, the speed distribution across torrents is a bimodal—the slowest (between 0 and 50 kB/s) and the fastest (between 350 and 400 kB/s) downloads together make 42% of all. Studying absolute differences for data with bimodal distribution can lead to wrong conclusions. For example, 10 kB/s speedup is crucial if it changes the speed from 10 kB/s to 20 kB/s and almost negligible if it changes the speed from 370 kB/s to 380 kB/s.

To better analyze the speed change, we define a new metric *speed diff*: it describes how much the speed of a download has changed. For a measured speed $speed_m$ and reference speed $speed_{ref}$ we calculate the *speed diff* as:

$(speed_m - speed_{ref})/min(speed_m, s_{ref})$

For example, if the reference download speed (0%-locality) is 100 kB/s and the measured speed for 33%-locality is 157 kB/s then the *speed diff* value is 0.57. And if the download speed for 67%-locality is 45 kB/s the *speed diff* is -1.22. In the example from the previous paragraph: *speed diff* between 10 kB/s and 20 kB/s is 1.00 but *speed diff* between 370 kB/s and 380 kB/s is only around 0.03.

In contrast to absolute speed, *speed diff* has a distribution very close to normal and in our opinion *speed diff* is a more appropriate metric for studying the change in download speed. For the sake of comparison with other locality studies, we do the statistical analysis for both *speed diff* and absolute speed in our results.

7.4 Statistical Model

We use a General Linear Model (GLM) to measure locality influence on download speed. We build a multiple linear regression model with main effects of nominal independent variables. It allows us to estimate the influence of a nominal variable X on a scale variable Y for every value of the nominal variable X in relation to a chosen value of reference—how every level of locality affects download speed in relation to the lack of locality (0%-locality).

We use two different dependent variables: download speed and *speed diff*. The first refers to absolute speed change, while the other to the relative speed difference. While speed has bimodal distribution, *speed diff* distribution is much closer to normal distribution which makes the mean of this variable a more convenient and reliable statistical parameter.

The only independent variable is locality entered as a 4-value nominal variable (0, 33, 67 and 100 with 0 as the level of reference). We analyze data split by location (the relationship between locality and dependent variable tested in every location separately) and the whole data (all locations together).

Remember that every swarm was joined four times, at the same time and with the same conditions. Hence, we exclude the effect of variables like number of seeders, numbers of leechers and file size.

We assume a confidence level of 0.95 in the analysis.

7.5 Average speed analysis

Our model revealed a statistically significant relationship between locality and download speed in four out of the five vantage points. However, the relationship is different, depending on the vantage point. Figure 6 presents average changes for both *speed diff* and absolute speed difference, together with 95% confidence intervals. While the results differ between metrics, we consider the difference to be not essential. We concentrate our analysis on *speed diff*, as it is the metric closer to the end-user experience.

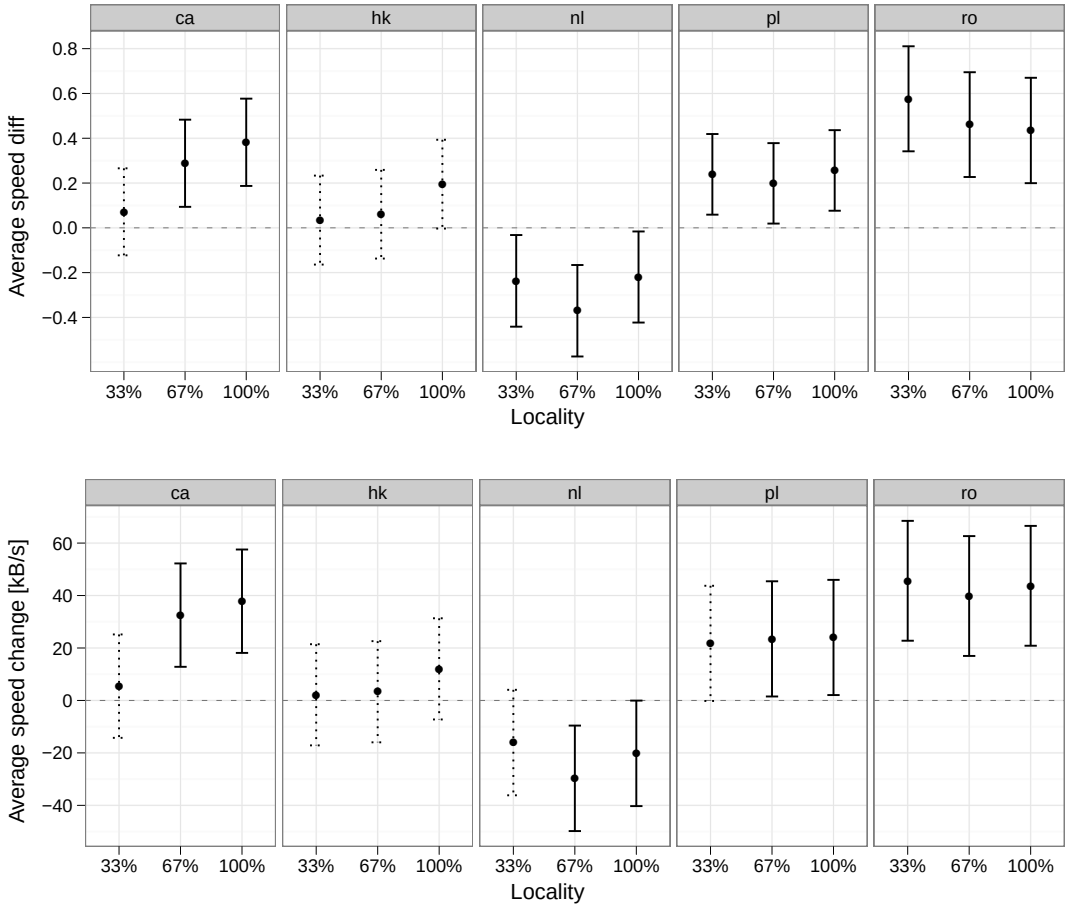


Figure 6: Mean speed diff *and absolute speed change, together with 95% confidence interval. Statically insignificant results drawn using dashed line. Poland, Canada and Romania notice significant performance increase, while performance for Hong Kong does not yield statistically significant results. A clear performance decrease is seen in the Netherlands.*

In Romania, we observe highest average download speed increase (up to 95% CI of 57.61% ± 23). This matches results obtained by Choffnes et al. [3] and can be explained by the nature of the connectivity in Romania. Namely, many ISPs in Romania provide higher bandwidth for peers inside the ISP or country, than to the outside world. Locality-based peer-selection results in higher utilization of faster local links, improving the overall download performance.

In Poland we also observe significant speed increase for all levels of locality.

Measurements for Canada are the most heterogeneous. For 67%- and 100%-locality we notice clear performance increase (up to 95% CI of 38% ± 20). However, there is no performance increase for 33%-locality. We cannot explain this phenomenon using our data and consider it to be a good example of how locality can lead to non-trivial results in real-world settings.

We do not notice any significant performance change in Hong Kong. This can be explained by the observation that this location has much higher median latency values and very high kilometers per packet metric, as shown in Section 7.1. Hong Kong is a location with too few

local peers to benefit from our locality solution.

Finally, we find that locality implies significant performance decrease in the Netherlands. However, the explanation that applies to Hong Kong does not seem to be reasonable for the Netherlands. In contrast to Hong Kong, this location has very good median latency to BGP prefixes and enjoys the best average kilometers per packet value (855km for 100%-locality level). In fact, it seems that, out of all five locations, the one in Netherlands has the best connectivity.

We observe a large diversity of results depending on location: using only five vantage points, we manage to observe positive, neutral and negative influence of locality on download speed. While positive influence is dominating, the two other cases are clear examples that locality is not always beneficial for the end-user.

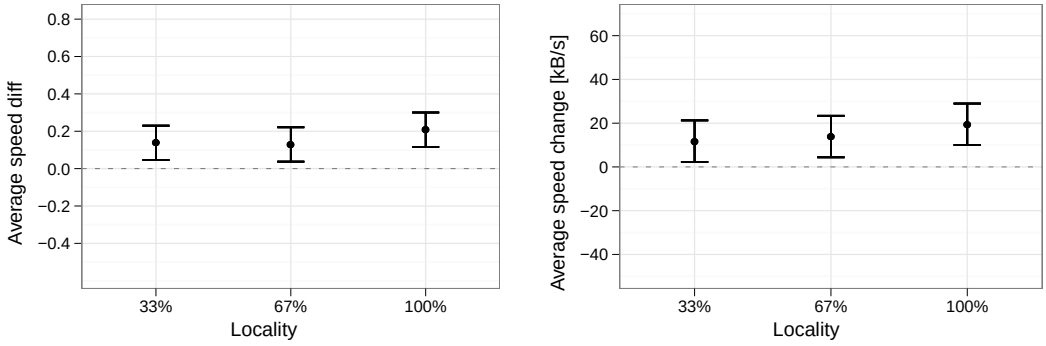


Figure 7: *Mean* speed diff *and absolute speed change for all locations together. Looking at our results from this perspective can give a false impression that locality always has statistically significant influence on download speed.*

Besides studying each location separately, we also look at all locations together. While our vantage point selection is not representative of the whole Internet, not differentiating between network locations is a common practice in P2P-locality publications [3] [10]. Studying all locations together is similar to answering the question: *Does locality increase the download speed in general?* The average changes, for both speed metrics, are presented in Figure 7. Surprisingly, the global picture masks the diversity of results for particular locations: we notice statistically significant performance increase for all levels of locality (up to 95% CI of 20.8% ± 9). This means that if we did not study all locations separately, we could remain unaware of the fact that in some conditions locality decreases end-user performance.

We are unable to say whether locality deployment should be deployed in the current Internet. We believe locality has the potential to increase the average download speed, but it does not mean that locality will increase the speed for every user. The neutral and negative influence of traffic localization needs to be better understood before deploying an actual solution.

In our opinion the discrepancy between the results for all locations together and results for specific locations has some serious implications. Firstly, it shows that current P2P-locality simulations do not capture the heterogeneity of the Internet topology. Secondly, the diversity of the results shows that it is not enough to analyze the locality in general. It is a case where better on average does not imply better for everyone. Lastly, it shows that there are real-world cases where locality leads to significant performance decrease. Those cases need to be studied more carefully before any locality-based solutions are deployed.

One example how heterogeneity of the Internet affects locality is the difference between the Netherlands and Romania. According to our best knowledge, the Netherlands is a country with

very good backbone connectivity, but with many end-users having ADSL or Cable connection. Lack of backbone bottlenecks could be a good explanation why locality does not have positive influence on download speed there. In contrast to the Netherlands, Romania has a rather poor outside connectivity, but many end-users have Ethernet connection at home, with fiber to the premise. According to the "State of the Internet Q4 2009" [18] report by Akamai, Romania has the third fastest broadband connectivity, with 45% of users having connections above 5 Mbit/s. In the Netherlands it is only 28%. In such a scenario, locality thrives because it can omit the bottlenecks of the outside connectivity and utilize a very good local connectivity. Again, it might be very hard to develop a good P2P-locality solution until we better understand these kinds of peculiarities.

8 Conclussions and thoughts for future work

In this paper we presented the design and results of a real-world P2P-locality measurement. We have used a latency-based approach for achieving locality and fed local peers to a regular BitTorrent client. Using our setup, we downloaded 6260 torrents in five locations (Canada, Hong Kong, the Netherlands, Poland and Romania) on 3 continents and applied statistical analysis on the results.

Our measurements revealed three major problems with P2P-locality:

- There are real-world conditions where it leads to significant performance decrease (as exemplified by the Netherlands).
- There is only limited potential for locality in some locations. In our case this is Hong Kong, but we suspect this is typical for many locations that are far from Europe and North America.
- Looking only at the global picture masks the diversity of the results in particular locations.

Our measurements have some limitations: we use only five locations, with particular content and client settings. In our opinion, the diversity of results we have obtained shows the need for more large-scale tests, similar to those performed by Choffnes et al. [3]. P2P-locality needs to be evaluated in more locations, with more diversity in both content and connectivity, and with various speed settings. Also, more solutions need to be tested in real-world conditions. In addition to that, more metrics should be studied, like for example: download time, maximum achieved slowdown, upload speed, median download time or global swarm speed.

We also need to better understand the influence the level of locality has on the download speed. We do not observe a difference between locality levels in four locations but in Canada the lowest (33%) locality level does not result in performance change. There is a tradeoff between too little bias towards local peers, resulting in no speed change, and too large bias, possibly resulting in peer clustering. Le Blond et al. [19] have argued that even high locality levels do not decrease the swarm performance, but this has yet to be measured in real-world conditions. Note that measurement like ours cannot reveal problems like peer clustering—global experiments are necessary.

The above-mentioned factors are often studied using simulations. However, using only five locations we achieve higher results diversity than most of the simulation-based studies. In our opinion, many P2P-locality simulations need to revisited, taking more parameters into account. Insights from simulations, and their underlying simplifying assumptions, need to be validated against real-world data.

The IETF ALTO working group [7] is currently designing a protocol for topology information exchange between ISPs and P2P users. This type of approach has yet to be tested in the real-world and we think that simulations should not be used as the final evaluation tool. We hope to see real-world measurements concerning ISP-based approaches in the near future. Also, our measurements strongly suggest it is crucial for those measurements to take many border cases into account.

Finally, we believe, we are still at the beginning of understanding how locality affects the end-user performance in the real-world. We hope our work will increase the awareness that locality is not always a clear win-win situation for ISPs and P2P users.

9 Data availability

Data from the measurements is available on our website (http://www.pds.ewi.tudelft.nl/~maciek/locality/). This is the data concerning swarm sizes, physical locations of peers, and download progress, used to generate figures 4, 5, 6 and 7.

10 Acknowledgments

We would like to thank Adam Osuchowski, Piotr Strzyżewski and Răzvan Deaconescu for providing us the infrastructure to do the measurements. We would also like to thank Oskar Żyndul and Agnieszka Sowa for assistance with statistical analysis and Mihai Capotă, Nazareno Andrade, David Hales and Elisa Jasińska for their valuable help on various stages of the draft.

This work is part of the P2P-NEXT project, supported by the European Commission through FP7 grant no 216217, http://p2p-next.eu.

References

[1] B. Cohen, *Incentives build robustness of BitTorrent.*, In Proceedings of First Workshop of Economics of Peer-to-Peer Systems, USA 2003. 4

[2] A. Parker, *The True Picture of Peer-To-Peer File-Sharing*, Panel Presentation, IEEE 10th International Workshop on Web Content Caching and Distribution, Sophia Antipolis, France, 2005 4

[3] D. R. Choffnes and F. E. Bustamante, *Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems*, In Proceedings of SIGCOMM 2008 4, 5, 7, 17, 18, 19

[4] M. Slot and P. Costa and G. Pierre and V. Rai, *Zero-Day Reconciliation of BitTorrent Users with Their ISPs*, Euro-Par 2009 4

[5] V. Aggarwal and A. Feldmann and C. Scheideler, *Can ISPS and P2P users cooperate for improved performance?*, In Proceedings of SIGCOMM, 2007 4, 5

[6] H. Xie and Y. R. Yang and A. Krishnamurthy and Y. G. Liu and A. Silberschatz, *P4p: provider portal for applications*, In Proceedings of SIGCOMM, 2008 4, 5

[7] E. Marocco and V. Gurbani, *Application-Layer Traffic Optimization (ALTO) Problem Statement.* ID, draft-marocco-alto-problem-statement-03 (Work in Progress), May 2009. 4, 5, 20

[8] R. Bindal and P. Cao and W. Chan and J. Medved and G. Suvala and T. Bates and A. Zhang *Improving Traffic Locality in BitTorrent via Biased Neighbor Selection* Proceedings of the 26th IEEE ICDCS, 2006 5

[9] E. Marocco and A. Fusca and I. Rimac and V. Gurbani, *Mythbustering Peer-to-peer Traffic Localization* ID, draft-marocco-p2prg-mythbustering-01, July 2009 5

[10] C. Griffiths and J. Livingwood and L. Popkin and R. Woundy and Y. Yang, *Comcast's ISP Experiences In a P4P Technical Trial* ID, draft-livingood-woundy-p4p-experiences-10 5, 18

[11] *Pando*, http://www.pando.com/ 5

[12] *CAIDA*, http://www.caida.org/home/ 9

[13] *MaxMind*, http://www.maxmind.com/ 10

[14] *The libTorrent and rTorrent Project*, http://libtorrent.rakshasa.no/ 12, 13

[15] B. Li and Y. Qu and Y. Keung and S. Xie and C. Lin and J. Liu and X. Zhang, *Inside the New Coolstreaming: Principles, Measurements and Performance Implications*, In Proceedings of INFOCOM 2008 13

[16] R. Cuevas and N. Laoutaris and X. Yang and G. Siganos and P. Rodriguez, *Deep Diving into BitTorrent Locality*, arXiv:0907.3874v2 13

[17] *Mininova*, http://www.mininova.org/ 14

[18] *State of the Internet Q4 2008*, Akamai, http://www.akamai.com/stateoftheinternet/ 19

[19] S. Le Blond and A. Legout and W. Dabbous. *Pushing bittorrent locality to the limit.* Technical Report inria-00343822, version 1 - December 2008, INRIA 19