Pushing the Performance of Biased Neighbor Selection through Biased Unchoking

Simon Oechsner, Frank Lehrieder, Tobias Hoßfeld, Florian Metzger, Dirk Staehle
Chair of Distributed Systems, Department of Computer Science, University of Würzburg
oechsner@informatik.uni-wuerzburg.de

Konstantin Pussep
Multimedia Communications Lab, Technische Universität Darmstadt, Germany
Konstantin.Pussep@kom.tu-darmstadt.de

Abstract

Locality promotion in P2P content distribution networks is currently a major research topic. One of the goals of all discussed approaches is to reduce the inter-domain traffic that causes high costs for ISPs. However, the focus of the work in this field is generally on the type of locality information that is provided to the overlay and on the entities that exchange this information. An aspect that is mostly neglected is how this information is used by the peers. In this paper, we consider the predominant approach of Biased Neighbor Selection and compare it with Biased Unchoking, which is an alternative locality aware peer selection strategy that we propose in this paper. We show that both mechanisms complement each other for the BitTorrent file sharing application and achieve the best performance when combined.

1. Introduction

Peer-to-peer (P2P) networks like, e.g., BitTorrent [2, 12] are widely used in today's Internet for content distribution. Compared to the client-server architecture, they offer significant advantages for the content provider. Since all peers interested in the content contribute resources, i.e., storage and upload capacity, to the distribution process of the content, scalability issues do not arise. Instead, the upload capacity of a P2P based content distribution network (CDN) increases with the number of users (peers) interested in a specific content. However, most P2P CDNs are oblivious of the underlying network topology and peers choose the sources of their downloads just according to overlay metrics or even randomly. This poses a major challenge for Internet Service Providers (ISPs) as it makes traffic engineering difficult if not impossible. The fact that such P2P based CDNs produce a large portion of today's Internet traffic [19] makes this even more problematic for ISPs as they are mostly charged on basis of the amount of traffic they send to or receive from neighboring autonomous systems (ASes).

Several different attempts have been made to address this problem. Some ISPs shut down connections they identified as BitTorrent connections leaving their AS or throttle the bandwidth of these connections [10]. In contrast to these unilateral means of the ISP to reduce inter-AS traffic, a lot of current P2P research focuses on a cooperation between the ISPs and the content providers [10, 9, 7, 20]. This goes as far as aiming for a triple-win situation, where ISPs, overlay providers and users all profit from a joint effort at traffic management, such as described in the SmoothIT project [13].

In these approaches, the ISPs provide information about the network topology to the overlay application, e.g., which peers reside in the same AS and which not. The peers use this information and communicate preferentially with peers in the same AS. This is also the main scenario discussed by the recently established IETF working group on application layer traffic optimization (ALTO) [1]. In [11], the authors present an approach that uses CDN servers like, e.g., Akamai as landmarks and does therefore not rely on an ISP provided service for the localization of other peers.

As in the given example, the most common form of topology information is locality awareness. Peers are considered local if they are close in the network topol-

This work has been performed in the framework of the EU ICT Project SmoothIT (FP7-2007-ICT-216259). The authors alone are responsible for the content of the paper.

ogy, typically expressed by a low latency or a low number of AS hops. Other forms of topology information may relate to high-bandwidth connections or belonging to the same VPN. In this paper, we focus on locality awareness in terms of AS hops, with local peers being defined as peers in the same AS and remote peers being peers in other ASes if not stated otherwise.

Locality-aware peer behavior consists of two steps. First, the peers need the information which other peers are local and which are remote. Second, the peers have to integrate that information when communicating with other peers.

The primary strategy recently investigated for this in the literature is Biased Neighbor Selection (BNS), which was introduced in [9] and adapted or enhanced in [7, 20, 11]. To complement this, we introduce the concept of Biased Unchoking (BU) in this paper and compare it to BNS.

In this study, we take the locality information as given, i.e., all peers know whether another peer is in the same AS or not, and focus on the evaluation of different possibilities to use that information. To this end, we analyze a BitTorrent based file-sharing network and modify the neighbor selection as well as the unchoking mechanism to incorporate the locality information. We will present an implementation of BU and compare its performance to the one of BNS in various scenarios.

The paper is structured as follows: Section 2 gives an overview on the existing approaches to BNS and other mechanisms to promote locality in CDN overlays. BitTorrent and the mechanisms for BNS and BU in these overlays are described in Section 3, while Section 4 contains the simulation results of the performance evaluation. Finally, we conclude the paper in Section 5.

2. Related Work

In the following, we give a short overview of different proposals for locality promotion in P2P CDNs. Some of them do not require adaptations of the P2P protocol, others do so. Since this paper focuses on the design options for the adaptations of a P2P application, we present these approaches in more detail.

2.1. Locality Promotion Strategies Without P2P Adaptations

Regarding the management of P2P traffic, the main goal of ISPs is to reduce the costly interdomain traffic. To this end, they can pursue different strategies. One approach is that the ISP closes connections to peers in other ASes. However, this led to dissatisfied users in the case of Comcast [6]. A similar idea is bandwidth throttling of P2P inter-AS connections. Both approaches have in common that P2P connections must be identified and the user experience will degrade.

Another option for ISPs to reduce the interdomain traffic is to introduce caches [15, 18]. These caches store popular content and redistribute it to local peers. However, the caches need to be tailored to the P2P application protocol in order to communicate with the peers. Furthermore, there may be legal issues when copyright protected content is cached by an ISP.

2.2. P2P Adaptations

Bindal et al. introduced the concept of BNS, where the neighbor set of a peer is changed so that it contains a significant fraction of peers that are close in terms of network proximity [9]. The most extreme case featured only one remote neighbor per peer, with the rest being local peers. It was found that inter-AS traffic can be reduced significantly, while the download times of peers are not influenced much in topologies where the access bandwidth of the peers is the bottleneck. In this approach, it was envisioned that either the tracker responds to queries with the target number of local peers, or that connections are artificially established or reset by traffic shaping devices of providers.

Aggarwal et al. present an approach where the locality information is queried from an Oracle service [7]. A peer essentially asks this service which peer from a list of potential neighbors it should connect to. Underlay information is available at the oracle server so that it can respond with an optimized choice. Afterwards, the querying peer then conducts BNS again by establishing an overlay connection just with the recommended peer. The mechanism was evaluated for a Gnutella network, showing that the properties of the overlay graph, such as connectivity or diameter, are not negatively influenced. However, most of the evaluations consider the search aspect of the Gnutella network, for which the overlay graph is actually used. Still, one result shows that the share of intra-AS file exchange connections was increased to up to 40% in the observed scenarios.

The plugin Ono for the BitTorrent client Azureus/Vuze, presented in [11] by Choffnes and Bustamante, offers an alternative to a provider-assisted locality service by re-using available information from CDN name resolutions. The proximity of peers is judged by the CDN servers that the peers are resolved to. Since this resolution is influenced by the CDN provider using underlay information to assign favorable servers to a client, the resulting recommendation is also valid for overlay connections. Again, the peers then use these recommendations to conduct BNS, by making sure that Ono-suggested connections are kept in the neighbor set of a peer.

Measurements from clients using the Ono plugin show that the biased connections established follow shorter paths w.r.t. AS hops. Also, in case a provider offers higher bandwidth to intra-network connections than to connections leaving the network, the download rates of peers using Ono improve significantly. In networks where the bottleneck is the access network, however, no great improvements, but also no large negative impact on the download performance was seen.

The approach followed by Xie et al. [20] in the P4P project is somewhat similar to the Oracle service in that an information server is used to offer underlay information to the overlay. Here, these entities are called iTracker, which may communicate with peers themselves or application trackers such as the BitTorrent tracker. In the evaluated scenarios, the communication took place between iTracker and application trackers. The evaluations range from simulations to measurements in PlanetLab and real CDN networks and show a significant reduction in inter-AS and bottleneck traffic with according iTracker optimization settings, while download times are slightly reduced in general.

3. BitTorrent and Locality-Aware Peer Selection Algorithms

In this section, we first describe the key mechanisms of BitTorrent because BitTorrent will serve as the example overlay application to illustrate BNS and BU. Furthermore, the simulation study in Sect. 4 is based on BitTorrent. Then, we describe the information exchange between the overlay and the underlay network. Finally, we present BNS and BU in detail. Both of them are forms of locality-aware peer selection algorithms. We explain their functionality and describe the required adaptations of BitTorrent to support them.

3.1. Key Mechanisms of BitTorrent

BitTorrent is a P2P content distribution protocol that offers multi-source download functionality. One overlay, also called *swarm* in BitTorrent terminology, is formed per file that is shared. A file is separated into smaller pieces called chunks to facilitate the fast generation of new sources. Each chunk is again divided into smaller subpieces, so-called blocks. A detailed description of BitTorrent can be found in [16] and [3]. In the following, we focus on the key mechanisms for traffic optimization, which are the management of the neighbor set and the choke algorithm.

3.1.1. Neighbor Set Management. Peers join a swarm by contacting a *tracker*, which is an index server holding information about all peers participating in the swarm. The address of the tracker itself is usually obtained from a website together with some information about the file, in the form of a .torrent file. When a peer contacts the tracker, it is supplied with an initial set of addresses of other peers in the swarm. In a standard tracker implementation, the peers returned to a requesting peer are random, with no filtering in terms of locality. Then, the joining peer tries to establish a neighborship relation to the peers it got from the tracker and collects all peers which accepted the request in his neighbor set.

After a neighborship relation is established between two peers A and B, they exchange information about which chunks of the file they have already downloaded and which not. If peer B has some chunks which peer A still needs to download, then peer A signals its interest to peer B and we say that peer A is *interested* in peer B.

3.1.2. Choke Algorithm. The choke algorithm decides to which neighbor a peer is willing to upload data. These neighbors are called *unchoked* whereas the neighbors which do not receive data are called *choked*. Every 10 seconds, the peer unchokes a default number of 3 of its interested neighbors. Which neighbors are unchoked depends on whether the peer has already downloaded the complete file (seeder mode) or not (leecher mode). In leecher mode, the peer unchokes those 3 neighbors from which it receives the highest download rates. This strategy is called *tit-for-tat* and provides an incentive for peers to contribute upload bandwidth to the swarm.

In seeder mode, the peer keeps those 3 peers unchoked which were most recently unchoked. In both seeder and leecher mode, every 30 seconds one of the interested and choked neighbors is selected randomly and unchoked for the following 30 seconds. This is called optimistic unchoking and allows the peers to get to know new mutually beneficial connections. In combination with the optimistic unchoking, the choke algorithm in seeder state ensures that every 30 seconds, the peer with the longest unchoke time is choked and a new interested peer is unchoked. Consequently, the upload slots of a seeder are distributed in a fair way among its interested neighbors.

3.2. Information Exchange Between Overlay and Underlay

Both BNS and BU rely on some kind of information about the underlying network topology. This information can be retrieved from a special server such as the SmoothIT Information Service [13] or the iTracker [20]. This service assigns a locality value $L(x, y)$ to every pair of peer addresses x and y. For the calculation of $L(x, y)$, different strategies are proposed, e.g., in [20]. For example, the number of IP- or AS-hops between x and y can be used. More complex strategies are also possible, including traffic engineering preferences of the ISPs or dynamic load information from the network.

As an alternative to a central service, the locality values could also be calculated at the peers themselves using their own measurements. This is proposed and evaluated in [11], where CDN servers are used as landmarks. However, it is not possible to include traffic engineering preferences of the ISPs in the locality values $L(x, y)$ with this approach.

For the performance evaluation in Section 4, the locality value $L(x, y)$ represents the number of AS-hops on the path from address x to address y. Furthermore, we assume that this value is known at the peers and at the tracker. In practice, $L(x, y)$ can be obtained from an information service as described above.

3.3. Biased Neighbor Selection

BNS is a mechanism applicable to most overlays and different forms of it are proposed in [9, 7, 11, 20]. It tries to influence the composition of the neighbor set of a peer at address x taking into account the locality values $L(x, y)$ of potential neighbors with address y. Basically, there are two alternatives to implement this in a BitTorrent overlay, the tracker-based BNS and the peer-based BNS. In this paper, we focus on the first alternative, which we describe in the following.

When a peer at address x requests addresses y of other peers in the same swarm, the tracker uses the information contained in $L(x, y)$ when it creates the response. To that end, it queries an information service as described above when a peer query from x is received. In our performance evaluation (cf. Sect. 4), we use the AS-hop metric for $L(x, y)$. This means that when a peer at address x requests addresses of other peers in the swarm, the tracker tries to include a certain fraction $l \in [0; 1]$ of peers with $L(x, y) = 0$ in the response, i.e., peers that are in the same AS. If the swarm does not contain sufficient peers in the AS of x, the tracker fills the response with other peers to avoid a degeneration of the connectivity of the overlay.

3.4. Biased Unchoking

With BU, a peer preferentially exchanges data with neighbors with a “good” locality value $L(x, y)$. This mechanism is generally applicable in content distribution overlays where peers have to compete for the upload bandwidth of a source. This is the case for BitTorrent or eDonkey, where the number of upload slots is normally limited. However, we focus on the description of concrete implementations of BU in BitTorrent. Here, the choke algorithm selects the neighbors to which a peer allocates its upload capacity. Consequently, BU influences the choke algorithm. In contrast, BNS influences the neighbor set management.

BU is motivated by the fact that – apart from the composition of the neighbor set – the choke algorithm has a major impact on which peers exchange data and how much. Especially, when only a few peers in one AS are online, all that BNS can achieve is that these peers are in the neighbor set of each other. Still, the number of these neighbors may be small compared to the number of all neighbors and that constrains the performance of BNS. Therefore, we propose BU which is intended to boost the data exchange between peers in the same AS in those situations.

BU works as follows. With BitTorrent, all n interested and choked neighbors Y of a peer are selected to be optimistically unchoked with same probability $p_{ou}(Y) = 1/n$ (cf. Sect. 3.1.2). With BU, this probability $p_{ou}(Y, L(x, y))$ that an interested and choked neighbor with address y is selected to be optimistically unchoked by the peer at address x depends on the locality value $L(x, y)$. In this way, we can achieve that neighbors with “good” locality values are optimistically unchoked more often than other ones.

In our approach, we define a threshold T and divide the set of candidates in a set of preferred ones with $L(x, y) \leq T$ and the rest with $L(x, y) > T$. Then, with probability q the peer to be optimistically unchoked is chosen from the set of preferred peers and with probability $1 - q$ from the rest. If one of the two sets is empty, the peer is chosen from the other set.

Like for BNS, we use the AS-hop metric for $L(x, y)$ and set the threshold $T = 0$. That means that only peers located in the same AS are in the set of preferred peers. If this set is not empty, the peer to be optimistically unchoked is deterministically taken from this set, i.e., we set $q = 1$.

4. Performance Evaluation

The performance evaluation on the different locality-aware peer selection mechanisms was conducted by means of simulation. We first present the simulation model and a small measurement study which serves to motivate the chosen parameters for the simulation. Then, we describe the used simulator. Finally, we evaluate and compare the performance of BNS and BU in different scenarios.

4.1. Simulation Model

We simulate one BitTorrent swarm which exchanges a file of size 154.6 MB generated from an example TV show of about 21 minutes in medium quality. The file is divided into chunks of 512 KB and every chunk into blocks of 16 KB.

At simulation start, the swarm contains only the initial seed (content provider) which has the complete file. New peers join the swarm with an exponentially distributed inter-arrival time A with a mean value of $E[A] = 10$ s. They stay online until they downloaded the entire file plus an additional, exponentially distributed seeding time. If not mentioned differently, the mean seeding time is 10 minutes. There are no offline times during the lifetime of a peer. This resulted in a mean number of concurrently online peers between 120 and 200 depending on the simulated scenario. We simulate the swarm for 6.5 hours. As a result, one simulation run consists of about 2300 downloads. After an analysis of the data, we observed that the initial warm-up phase took about 1.5 hours in all of the simulation runs, which we discard for the evaluation.

The multi-AS underlay network we simulate is composed of one transit-AS and a number of stub-ASes connected via inter-AS links in a star topology, i.e., the stub-ASes are all connected to the transit-AS but not directly interconnected with each other. We use this simple setup since our evaluated mechanisms only differentiate between peers in the same AS and remote peers. A more complex topology would not have any effect on the observed results.

If not mentioned differently, the number of stub-ASes is 20. The peer arrival process is equally distributed over all stub-ASes, i.e., when a new peer arrives, it randomly joins one of the stub-ASes. The transit-AS does not contain any peer. The peers are connected to their stub-AS with an access speed of 16 Mbit/s downstream and 1 Mbit/s upstream, which are typical values for the DSL access technology. The tracker and the initial seed are placed in one of the stub-ASes. The seed has a symmetric upload and download bandwidth of 10 Mbit/s, respectively. The access links of the peers and the inter-AS links can act as network bottlenecks, i.e, limit the bandwidth of their connections. If not stated otherwise, we model the inter-AS links as well dimensioned.

4.2. Reasoning for the Swarm Model

In order to justify our overlay setup, we present here a study of typical BitTorrent swarms, in particular the number of peers in a swarm and how many peers in a swarm belong to the same AS. The measurements were repeated with different sets of BitTorrent swarms in order to get a random snapshot of existing swarms in the Internet. During April 2009, we measured 126,050 swarms offering movie files with the .torrent files being gathered from the tracker provider mininova.

Fig. 1 shows results from this study. We investigate which fraction of a swarm typically belongs to one AS. We see a scatter plot over all swarms in the trace, with their average ratio of peers per AS plotted against the swarm sizes.

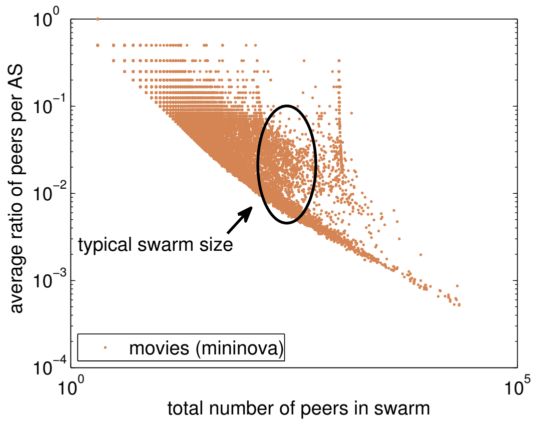

Figure 1. Scatterplot of the fraction of peers which belong to the same AS.

The study shows that for typical swarms, the number of peers in one AS is between 1% and 10%. Furthermore, more than 95 % of the normal swarms are smaller than 200 peers. The peer population tends to be split up in the network.[1] Therefore, we concentrate in this study on the common medium sized swarms and assume only small fractions of the swarm being in the same AS. Hence, we simulate a swarm being made up of small subgroups of close peers instead of fewer but larger local clusters. This is reflected in our model by having 20 stub-ASes and an average of 5% of all peers in one stub-AS by default.

4.3. Simulator

For our simulation studies, we used the P2P simulation and prototyping framework ProtoPeer [4] which is written in Java. We chose ProtoPeer mainly for two reasons: firstly, it already contains a network model for bandwidth-dependent overlay applications like BitTorrent. Secondly, it facilitates the development of overlay

[1] A more detailed paper on this study is currently under submission.

applications as only the specific peer behavior needs to be implemented within the framework.

For the underlay network, we use the flow-based network model provided by ProtoPeer. This network model mimics the property of TCP that the capacity of a link is shared among all data connections using this link. To simulate this bandwidth allocation, the bandwidth of the connections are assigned according to the max-min-fair-share principle [8]. The time a connection needs to transmit its data depends on the available bandwidth. When all data of a connection is transmitted, the connection is removed from the network. The use of such a flow-based network model for P2P simulations is proposed in [14] and [17]. These studies describe concrete implementations of the bandwidth allocation algorithms and evaluate their runtime speed. A comparison of the resulting transmission times to a packet-based NS-2 simulation is also given in [14].

While the network model was provided by ProtoPeer, the framework does not contain an implementation of the BitTorrent protocol. Therefore, we created a self-written implementation of BitTorrent according to the descriptions in [16] and [3]. It includes all key mechanisms, in particular the piece selection mechanisms, the management of the neighbor set, and the choke algorithm. Furthermore, the complete message exchange among the peers themselves, between peers and the tracker as well as between the peers and the information service for locality data is simulated in detail.

Since the bandwidth allocation process is a costly operation in terms of computation time, we only allocate bandwidth to connections which simulate the transmission of a block of the shared file from one peer to another. These are called piece messages in BitTorrent and have a size of 16 KB. All other messages in the BitTorrent protocol are orders of magnitude smaller than piece messages and therefore are assumed to have a negligible impact on the bandwidth dynamics of the network. To model the TCP handshake, we add a small, constant connection startup delay of 10 ms to the transmission of all messages.

4.4. Comparison of Biased Unchoking and Biased Neighbor Selection

BNS has been studied intensively [10, 9, 7, 20]. We include it in our performance evaluation to facilitate a comparison of BU and BNS. In all experiments, we compare 4 different peer behaviors: (1) regular BitTorrent (regBT), (2) BitTorrent with Biased Unchoking (BU), (3) BitTorrent with Biased Neighbor Selection (BNS), and (4) BitTorrent with both BNS and BU (BNS&BU). In all cases, the locality value $L(x, y)$ is the number of AS-hops on the path from x to y. We call a neighbor with address y of a peer with address x 'local' if $L(x, y) = 0$. With BU, a peer selects a local interested neighbor to be optimistically unchoked whenever possible (cf. Sect. 3.4). For BNS, we use the tracker-based version and set the fraction of local peers that the tracker tries to include in his response to $l = 0.9$ (cf. Sect. 3.3). This does not mean that 90% of the peers in every tracker response are in the same AS as the requesting peer because the number of those peers in the swarm is low in all our scenarios. In those cases, the tracker first includes the local peers in its response and then fills it with remote peers so that the number of peers in the tracker response is not affected.

In order to assess the performance from an ISP's perspective, we consider the amount of intra- and inter-AS traffic. This traffic was measured in intervals of one minute during the whole simulation and then averaged over one simulation run. If the source and the destination of a data transfer is in the same AS, the traffic is considered as intra-AS traffic. Otherwise, it contributes to the inter-AS traffic. To judge the overlay performance from the user's point of view, we consider the download times of the peer. This is the time when a peer issues its first block request until it has completely received the file. Here, we average the download times of all peers in one simulation run.

We run 10 simulations with different seeds and show average values over all runs for all observed variables. We calculated the confidence intervals for a confidence level of 95% but omit them in the figures for the sake of clarity. The size of all confidence intervals was below 7% of the corresponding mean value.

To compare BNS and BU, we consider different load scenarios and vary the fraction of peers which reside in the same AS. Furthermore, we investigate scenarios where the inter-AS links can be the network bottlenecks and where only some of the peers in the swarm are locality-aware.

4.4.1. Experiment "Load". In this experiment, we compare the performance of BNS and BU under different load conditions. Load here means the fraction of leechers in the swarm. To generate different load scenarios, we vary the mean seeding time of the peers from 5 to 30 minutes.

We start by taking a look at the inter-AS traffic. Figure 2 shows the mean value of this bandwidth for the different mechanisms and load scenarios. The scenario with 5 minutes mean seeding time is the one with the highest load. To judge the share of total traffic that is inter-AS traffic, the intra-AS traffic of every mechanism is also shown on top of the inter-AS traffic bars (labeled

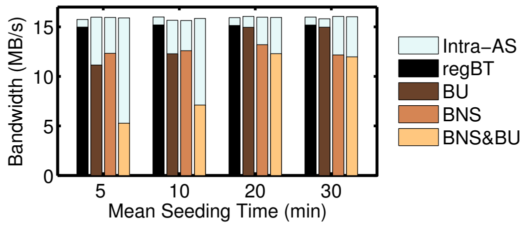

Figure 2. Mean inter-AS bandwidth between stub-ASes.

’Intra-AS’). Thus, the complete bar is the sum of both and therefore the total average bandwidth utilized.

The bandwidth of the total traffic for all mechanisms and scenarios is slightly below 16 MB/s which is equal to the generated traffic demand when on average every 10 seconds a peer joins the swarm and downloads the whole file of 154.6 MB. Furthermore, the inter-AS bandwidth of regBT is almost unaffected by varying mean seeding times. With regBT, only about 5% of the total traffic stays within the originating stub-AS. This corresponds exactly to the fraction of local neighbors of a peer (cf. Table 1). With BNS, a peer knows more local peers than with regBT and this reduces the inter-AS traffic. In all scenarios, with BNS the inter-AS traffic accounts for 75-80% of the total traffic.

With BU, the amount of inter-AS traffic is smaller for short seeding times. While the inter-AS traffic is reduced to about 11 MB/s in the scenario with 5 minutes mean seeding time, BU has almost no effect with 20 or 30 minutes mean seeding time. This is similar for the combination BNS&BU. For long mean seeding times, BNS&BU cannot save interdomain traffic. In contrast, it is especially effective in scenarios with short seeding times. There, only about 30% of the traffic are inter-AS traffic. The reason is that BNS takes care that a peer knows the other peers in the same AS while BU assures that these peers are unchoked whenever possible.

The fact BU and BNS&BU are more effective in scenarios with high load can be explained as follows. BU and also BNS&BU can only work when at least one local, interested, and choked neighbor exists in the neighbor set of a peer. Table 1 shows that this is only rarely the case in the scenarios with 20 or 30 minutes mean seeding time. Consequently, BU is effective when the load in the swarm is high, i.e., when peers have several interested neighbors. Then, it can select a local neighbor to be optimistically unchoked. This is in particular the case in so-called flash-crowd scenarios where many peers join the swarm in a short period of time. We tested this hypothesis but omit the figures due to the page limit.

This can also be observed in Figure 3, where the CDF of the average number of unchoke slots for local peers is plotted for two load scenarios, corresponding to 5 and 20 minutes mean seeding time. We can see that in the highly loaded system, BU and especially BNS&BU is able to give more unchoking slots to local peers than for a low load, although the number of peers in the local AS is the same. There seems to be a contradiction because BU only decides about one unchoking slot. However, optimistically unchoked peers may, by virtue of the tit-for-tat mechanism, be unchoked regularly after having been ’discovered’ via optimistic unchoking. In this manner, BU causes that all upload slots of a peer are preferentially allocated to local neighbors. Thereby, it does not devalue the tit-for-tat decision. It only suggests nearby neighbors for the optimistic unchoking. If they do not provide sufficient upload capacity to get one of the regular unchoke slots, they will be choked again.

Seeding time (min)	Interested Neighbors 5	10	20	30
regBT	29.06	19.64	4.51	2.21
BU	29.20	19.22	4.56	2.27
BNS	29.16	19.53	4.83	2.21
BNS&BU	29.19	19.35	4.76	2.24
Seeding time (min)	**Local Neighbors 5**	**10**	**20**	**30**
regBT	2.13	2.15	2.14	2.14
BU	2.22	2.22	2.16	2.14
BNS	6.38	6.30	7.18	9.95
BNS&BU	6.61	6.46	7.10	9.89
Seeding time (min)	**Local Interested Neighbors 5**	**10**	**20**	**30**
regBT	1.43	0.97	0.23	0.11
BU	1.39	0.93	0.23	0.11
BNS	**4.31**	**2.88**	0.84	0.52
BNS&BU	**4.17**	**2.78**	0.80	0.53

Table 1. Mean number of different types of neighbors of a peer. The mean number of total neighbors was about 43 in all cases.

Finally, we observe no large impact of the evaluated mechanisms on the mean download times of the file. These are 14.6, 9.9, 2.6, and 1.7 minutes in the scenarios with 5, 10, 20, and 30 minutes mean seeding time, respectively. They do not differ significantly (below 10s) among the investigated mechanisms. Therefore, we argue that a user will not see a big difference in the performance of the application, while the gains for an ISP are potentially large.

4.4.2. Experiment “Swarm Distribution”. Next, we evaluate the impact of the distribution of peers on ASes, since a smaller number of potential local neighbors

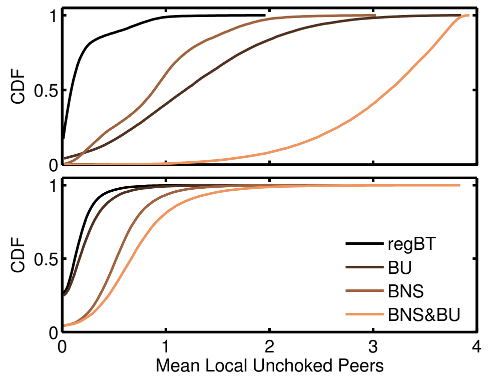

Figure 3. CDF of number of local unchoked peers for 5 (above) and 20 min. (below) mean seeding time.

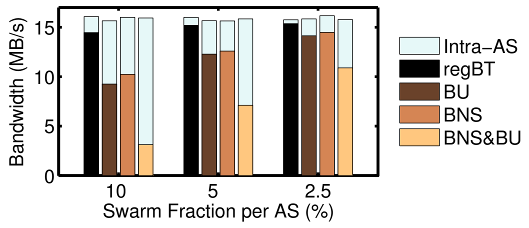

Figure 4. Mean inter-AS bandwidth for different swarm distributions.

means less opportunity to promote locality. To this end, we vary the number of stub-ASes in the simulated topology. Since a new peer appears in each stub-AS with equal probability, each stub-AS receives a smaller fraction of the swarm if there are more ASes. We simulate topologies with 10, 20 and 40 stub-ASes, resulting in 10%, 5%, and 2.5% of the swarm per AS on average.

Again, we take a look at the inter-AS bandwidth savings achieved by the different mechanisms, cf. Figure 4. In general, the gains made by all locality-promoting mechanisms are larger if the fraction of the swarm in one AS is large. BNS profits directly from more local peers since the share of local neighbors per peer is also higher. Also, BU has a higher probability to find a local interested neighbor when there are more peers in the same AS. The combination of both mechanisms utilizes both of these advantages, leading to an improvement from 30% saved inter-AS bandwidth with just BNS to close to 80% saved with the combination in the scenario with a share of 10% of the swarm per AS, both in relation to regular regBT.

The inter-AS traffic reduction is decreased when the local share of the swarm gets smaller. For the scenario with an average of 2.5% of the peers in one AS, BNS and BU alone save only in the range of 6% of the inter-AS traffic, while BNS and BU together still reduce the traffic of regular BitTorrent by 30%. The reason is that the combination of both mechanisms tries to utilize every last local neighbor. With BNS alone, the probability that a local neighbor is unchoked is small. With BU alone, the probability that a local peer is in the neighbor set is small. Consequently, they cannot reduce inter-AS traffic alone in scenarios where only a very small fraction of the peers resides in the same AS. However, when they are used in combination, the savings are up to 5 times higher than for both mechanisms alone. This shows that BU can push the performance of BNS in particular when only few local peers are in the swarm.

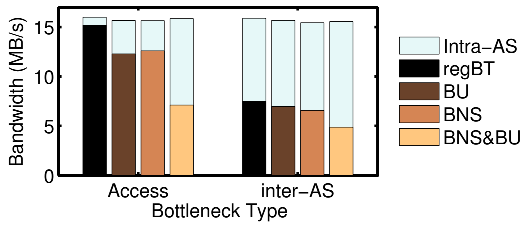

Figure 5. Mean inter-AS bandwidth with and without inter-AS bottlenecks.

Since we again have no bottleneck in the network, the location of neighbors does not have an effect on the utilized download bandwidth per peer. As a consequence, the download times are not affected by the number of stub-ASes nor by the different mechanisms. For all configurations, the mean download times are slightly below 10 minutes.

4.4.3. Experiment "Inter-AS Bottlenecks". In this section, we investigate the impact of "inter-AS bottlenecks", i.e., bandwidth limitations of the links between the stub-AS and the transit-AS. The experiment is motivated by the fact that some providers throttle the bandwidth of P2P connections leaving their network [11].

The authors of [11] show that under these conditions locality awareness leads to a better application performance since the bottleneck link is avoided and local connections with higher throughput are preferred. To judge whether BU also works well under these circumstances, we limit the capacity of each inter-AS link in our topology to 3072 kbit/s, i.e., three times the upload capacity of one peer. We compare the results to the scenario with no limitations on the inter-AS links, labeled 'Access bottleneck'.

The interdomain bottlenecks result in generally lower inter-AS bandwidths for all mechanisms (cf. Figure 5).

No more than 7.68 MB/s can be uploaded from all 20 ASes simultaneously, because every one of the 20 links from a stub-AS to the transit-AS has a capacity of only 3072 kbit/s. In contrast to regBT, the locality-aware mechanisms keep the interdomain traffic below that limit because inter-AS connections whose bandwidth is limited on an inter-AS link are likely to be replaced by the intra-AS connections with higher bandwidth. This is caused by the tit-for-tat policy of BitTorrent which allocates upload slots to those peers from which it gets the best download speed.

However, with inter-AS bottlenecks, the download times are no longer independent from the mechanism (cf. Figure 6), because different sources offer a different bandwidth for download. Thus, the download times for regBT are much longer than in the scenario where connections are limited only by the access links. Since here local peers with good connectivity may be discovered only via the regular unchoking process, many low-bandwidth connections via inter-AS links are utilized. The effective capacity of the system is reduced, leading to download times that are three times longer than without inter-AS bottlenecks.

The locality-aware mechanisms on the other hand foster the utilization of the better connectivity between local neighbors since these are preferred anyways. In our scenario, the combination of BU and BNS leads to only a slight increase in the mean download times compared to the scenario without inter-AS bottlenecks. This can be explained by the fact that the mean inter-AS bandwidth in the scenario without inter-AS bottlenecks was already below the capacity limit introduced by the inter-AS bottlenecks. Therefore, the performance of BNS&BU is only affected to a minor degree. The impact of the inter-AS bottlenecks is larger for BNS and BU alone. Still, the mean download times are considerably smaller than with regBT. From this experiment we conclude that in case of inter-AS bottlenecks, BU improves the mean download times compared to regBT and the combination of BNS&BU leads to shorter download times than BNS alone.

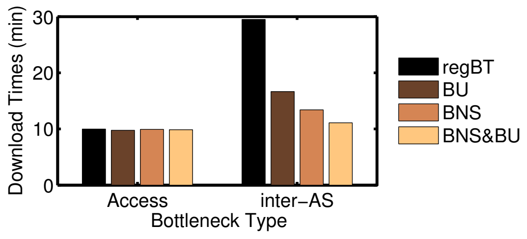

Figure 6. Mean download times with and without inter-AS bottlenecks.

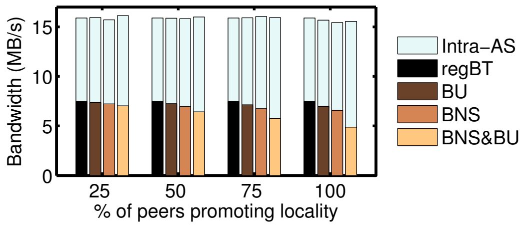

Figure 7. Mean inter-AS traffic for different shares of locality promoting peers in the swarm.

4.4.4. Experiment: "Fraction of Locality-Aware Peers". With the final experiment, we test what happens if only a fraction of the peers in the swarm promotes locality while the rest uses the standard BT implementation. We vary the share of peers that utilize a locality-aware mechanism from 0% (corresponding to the regBT case) to 100% (corresponding to the previous results). Here, we again simulate the 3 Mbit/s bottleneck in the inter-AS links.

The inter-AS traffic of the regBT implementation is again capped at the bandwidth limit introduced by the inter-AS bottleneck links (cf. Fig. 7). The locality-aware mechanisms save some of this inter-AS traffic even if only 25% of the peers actively promote locality. The savings increase with the share of peers utilizing locality-awareness. We also see that the addition of BU again enhances the BNS mechanism, since the combination of both leads to the largest savings.

As in the experiment before, the introduction of the inter-AS bottleneck has an impact on the download times of the peers, cf. Table 2. Here, we show the results separately for the two groups of peers, the ones that do support and the ones that do not. The locality-aware mechanisms all lead to shorter download times than the regular implementation for both groups. Even if only a fraction of the peers supports locality, it still helps the swarm by generating new sources faster and providing more upload bandwidth to the local neighbors of the locality-promoting peers. However, BU alone performs worst of the biased algorithms. Not only do the peers supporting BU experience the longest download times, they also do not improve their performance significantly over the peers that do not support locality.

In contrast, the peers implementing BNS and the combination of BNS and BU decrease their download times of the file by more than 50% in any scenario. They also perform better than the group ignoring locality, although this advantage diminishes when a larger part of the swarm is locality-aware. This again is due to the fact that regular peers also profit from the better performance

	Locality Promoting Peers			
Share (%)	25	50	75	100
regBT	-	-	-	-
BU	22.95	20.24	18.54	16.65
BNS	13.04	13.59	13.52	13.40
BNS&BU	10.90	11.39	11.14	11.09

	non-Locality Promoting Peers			
Share (%)	25	50	75	100
regBT	-	-	-	29.50
BU	24.85	21.41	18.94	29.50
BNS	22.25	18.43	16.41	29.50
BNS&BU	20.33	15.89	14.00	29.50

Table 2. Mean download times (in minutes) of locality promoting peers (top) and non-locality promoting peers (bottom).

of the locality-aware peers.

5. Conclusion

In this paper, we compared the standard BitTorrent implementation and a variant including Biased Neighbor Selection with our new approach of Biased Unchoking. We found that Biased Unchoking works best in scenarios with high load on the swarm. The combination of Biased Neighbor Selection with Biased Unchoking leads to the best performance in this comparison, since Biased Neighbor Selection provides the local neighbors that can then be preferred by Biased Unchoking. Thus, both mechanisms complement each other well and should be used together.

In scenarios with a bandwidth bottleneck in the inter-AS links, a combination of Biased Unchoking with Biased Neighbor Selection also leads to shorter download times, in accordance with earlier results showing that traffic locality profits more from interdomain link limits. Even if only a fraction of the peers in a swarm used the two mechanisms, the whole swarm as well as the underlay providers profit.

As future work, we will consider more complex topologies and locality metrics, as well as other applications, such as video-on-demand streaming as implemented in Tribler [5]. Also, scenarios with heterogeneous access speeds of the peers will be evaluated.

References

[1] Application-layer traffic optimization (alto). http://www.ietf.org/html.charters/alto-charter.html.
[2] Bittorrent. http://www.bittorrent.com/.
[3] Bittorrent specification. http://wiki.theory.org/BitTorrentSpecification.
[4] Protopeer. http://protopeer.epfl.ch/index.html.
[5] Tribler. http://www.tribler.org/.
[6] Comcast throttles bittorrent traffic, seeding impossible. http://torrentfreak.com/comcast-throttles-bittorrent-traffic-seeding-impossible/, 2007.
[7] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P systems co-operate for improved performance? *ACM SIGCOMM Computer Communications Review (CCR)*, 37(3):29–40, July 2007.
[8] D. Bertsekas and R. Gallagher. *Data Networks*. Prentice Hall, 1987.
[9] R. Bindal, P. Cao, W. Chan, J. Medval, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 66. IEEE, IEEE Computer Society Washington, DC, USA, 2006.
[10] S. L. Blond, A. Legout, and W. Dabbous. Pushing BitTorrent Locality to the Limit. Technical report, Dec 2008.
[11] D. R. Choffnes and F. E. Bustamante. Taming the torrent: a practical approach to reducing cross-ISP traffic in Peer-to-Peer systems. *SIGCOMM Comput. Commun. Rev.*, 38(4):363–374, 2008.
[12] B. Cohen. Bittorrent protocol specification, February 2005.
[13] J. P. Fernandez-Palacios Gimenez, M. A. Callejo Rodriguez, H. Hasan, T. Hoßfeld, D. Staehle, Z. Despotovic, W. Kellerer, K. Pussep, I. Papafili, G. D. Stamoulis, and B. Stiller. A New Approach for Managing Traffic of Overlay Applications of the SmoothIT Project. In *2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS '08)*, Bremen, Germany, July 2008.
[14] T. J. Giuli and M. Baker. Narses: A scalable flow-based network simulator. *CoRR*, cs.PF/0211024, 2002.
[15] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution? In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 63–76, Berkeley, CA, USA, 2005. USENIX Association.
[16] A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough, 2006.
[17] F. L. Piccolo, G. Bianchi, and S. Cassella. Efficient simulation of bandwidth allocation dynamics in p2p networks. In *Proceedings of the Global Telecommunications Conference, 2006. GLOBECOM '06, San Francisco, CA, USA*, 2006.
[18] O. Saleh and M. Hefeeda. Modeling and caching of peer-to-peer traffic. In *ICNP '06: Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.
[19] R. Steinmetz and K. Wehrle. *P2P Systems and Applications*. Springer Lecture Notes in Computer Science, 2005.
[20] H. Xie, R. Y. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz. P4P: Provider portal for applications. *SIGCOMM Comput. Commun. Rev.*, 38(4):351–362, 2008.