Performance Study of Locality-Aware Peer Selection Algorithms

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

Konstantin Pussep
Multimedia Communications Lab,
Technische Universität Darmstadt, Germany

***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 interdomain 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 generally neglected or only shortly addressed is how this information is used by the peers. In this paper, we compare the predominant approach of Biased Neighbor Selection and compare it with Biased Unchoking which is introduced in this paper.**

I. Introduction

Peer-to-peer (P2P) networks like, e.g., BitTorrent [1], [2] 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., 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 [3] makes this even more problematic for ISPs as they are mostly charged on basis of the amount of traffic they send to or received 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 from their AS to another one or throttled the bandwidth of these connections. While this reduces the costly inter-AS-traffic, it also compromises the quality of experience (QoE) of the users [4]. As a consequence, users become dissatisfied with the service of the ISP because file downloads take more time or video-on-demand (VoD) applications show the videos in an insufficient quality. 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 [5], [6], [7], [4]. 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) [8]. In [9], the authors present an approach that uses CDNs severs 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 awareness offered to the peers is locality awareness. Peers are considered local if they are close in the network topology, typically expressed by a low ping or a low number of AS hops. Other forms of topology-awareness may relate to high-bandwidth connections or belonging to the same VPN. In this paper, we will mainly 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 mainly 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 the Biased Neighbor Selection, which was introduced in [5] and adapted or enhanced in [6], [7], [9]. With biased neighbor selection, the peers try to adjust their set of neighbors so that it contains at least a certain fraction of local peers. This might be supported by the tracker if existing. However, since only the composition of the neighbor set is influenced by this strategy, there is no hard preference for local peers in the data exchange process. Therefore, we introduce the concept of a Biased Unchoking strategy in this paper and compare it to Biased Neighbor Selection. Biased Unchoking prefers local peers in the unchoking process implemented in BitTorrent-based overlays, and therefore has a direct influence on the establishment of data exchange connections.

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 several alternatives to implement Biased Unchoking and compare the performance of both strategies in various scenarios and show their advantages and drawbacks.

The paper is structured as follows: Section II gives an overview on the existing approaches to Biased Neighbor Selection and other mechanisms to promote locality in CDN overlays. BitTorrent, Tribler and the mechanisms for Biased Neighbor Selection and Biased Unchoking in these overlays are described in Section III, while Section IV contains the results of their performance evaluation. Finally, we conclude the paper in Section V.

II. 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 adaptions of the P2P protocol, others do so. Since this paper focuses on the design options for the adaptations of P2P application, we present these approaches in more detail.

A. Locality Promotion Strategies Without P2P Adaptations

Regarding the management of P2P traffic, the main goal of ISPs is to reduce the costly inter-domain 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 [x]. A similiar idea is bandwidth throttling of inter-AS peer connections. Both approaches have in common that P2P connections must be identified and the user experience may be degraded.

Another option for ISPs to reduce the inter-domain traffic is to introduce caches [10], [11]. These caches save 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 is cached by an ISP.

B. P2P Adaptations

Bindal et al. introduced the concept of Biased Neighbor Selection, 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 [5]. 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 re-routed by traffic shaping devices of providers.

Aggarwal et al. present an approach where the locality information is queried from an Oracle service instead [6]. 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 Biased Neighbor Selection again by establishing an overlay connection just with the recommended peer. The mechanism was evaluated with a Gnutella network, showing that the graph properties of the overlay graph are largely 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 inter-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 [9] by Choffnes and Bustamante, offers an alternative to a provider-assisted loclaity 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 Biased Neighbor Selection, 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, however, no great improvements, but also no large negative impact on the download performance was seen.

The approach followed by Xie et al. [7] in the P4P project is somewhat similar to the Oracle service in that a 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.

III. Locality-Aware Peer Selection Algorithms

In this section, we present different algorithms for peer selection which take into account the location of the neighbors of a peer. As we use BitTorrent and Tribler in this study, we first explain their key mechanisms. Then, we explain possible adaptations of these overlays in order to implement Biased Neighbor Selection or Biased Unchoking.

A. 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.

Peers join that swarm by contacting a *tracker*, which is basically an index server holding information about all peers participating in a 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. Once a peer joining the swarm has contacted the tracker, it is supplied with a number of initial contacts to establish connections to and to start exchanging data. In a standard tracker implementation, the contacts returned to a requesting peer are random, with no filtering in terms of locality.

All contacts a peer has connections with are his neighbors in the overlay. In this neighbor set, there are normally some peers that have already downloaded pieces of the shared file which the local peer still needs. The local peer signals his interest to download these pieces and is thus joining the set of interested peers at these neighbors.

The upload of data is managed by a so-called *unchoking* process. Every 10 seconds, a peer allocates upload slots to a default number of 4 interested peers. All other peers do not receive upload bandwidth from that peer and are therefore 'choked'. This part of the peer selection process involves a measure of the download bandwidth the local peer experiences from each of the candidates. The more an interested peer uploads to the local peer, the higher it is ranked. This *tit-for-tat* strategy provides an incentive for peers to contribute upload bandwidth to the swarm. To allow peers to get to know new mutually beneficial connections, an additional optimistic unchoking slot is given to a random interested peer every 30 seconds.

Once a peer is unchoked, it may select the piece of the file it wants to download. This is done according to a *least-shared first* or *rarest-first* metric, with variations at the beginning and the end of a file download. From the eligible pieces, i.e., the chunks that the downloader still needs and which the uploader has stored locally, the one that is seen the least by the downloader is selected. This mechanism is a countermeasure to the chunk starvation problem, where one or a small number of chunks is distributed much slower than the rest and may vanish completely from the swarm in the worst case.

B. Key Mechanisms of Tribler

Tribler is a BitTorrent-based Video-on-Demand streaming overlay that allows for watching a video while downloading it. It uses many of the basic mechanisms of BitTorrent, with the main differences being the peer and chunk selection algorithms.

While the basic unchoking procedure is the same as in BitTorrent, the metric used by the overlay to rank peers is not tit-for-tat, but give-to-get. This metric measures how much data from the local peer was forwarded by an interested peer, and how much data was uploaded by it in total in the last time interval. This is necessary due to the fact that peers tend to download the file roughly in order. Thus, peers starting to download and watch the video have little to offer to peers that are close to finishing their download, and therefore the reciprocity principle of tit-for-tat does not work efficiently.

The chunk selection of Tribler does not use the rarest-first strategy, but takes into account the current playout position of the local peer in the streamed video. It separates the remainder of the video, i.e., the parts being played out in the future, into three priority sets. Chunks in the first set are downloaded in order and with high priority, the chunks in the second and third set according to rarest-first and with medium and low priority, respectively. The in-order part of the strategy is used to minimize gaps in the received data during playback, which causes stall times to occur. If a peer is offered enough bandwidth and sources to download, the rarest-first part of the strategy tries to forestall chunk starvation, as in BitTorrent.

C. Biased Neighbor Selection

Biased Neighbor Selection is a basic mechanism applicable to most overlays. It tries to influence the composition of the neighbor set of a peer by preferring local neighbors to remote ones. A higher share of local neighbors then means a theoretically higher probability that such a neighbor is selected for data exchange. This is a desirable effect since local connections are much more resource-efficient and also mean lower costs for ISPs.

However, there are also some theoretical drawbacks to this method. First, if the overlay contains only a small number of peers that can be considered local, the share of local neighbors is small even if all of them can be found. Thus, they are in total also utilized less than remote peers which make up a larger share of the neighbor set. We will show in this paper that this is a realistic scenario in existing overlays.

Second, even if a significant share of neighbors is local, it is not guaranteed that they are selected for usage. Especially in the case of file sharing, the availability of pieces needed by the local peer governs which neighbors are potential uploaders and which do not have anything of interest. Thus, the actual traffic may not be distributed in the same way as the neighbor set is partitioned into local and remote peers.

There are two basic alternatives to implement Biased Neighbor Selection in a BitTorrent overlay. The first is letting the tracker choose a certain number of peers that are close to the requesting peer and including them in its response. This requires modifications at the tracker, which has to gather underlay information about the peers, e.g., by querying a special server such as the SmoothIT Information Service [12] or the iTracker [7]. The advantage of this approach is that all available local peers are considered when sending a peer its potential contact list, since the tracker knows all peers in a swarm. The drawbacks are a higher complexity at the tracker, which is already a critical component, as well as a lesser degree of freedom for the peers, which can not choose whether to actually support biased neighbor selection or not.

The second alternative remedies this by retaining the tracker functionality as it is and changing the peer behaviour instead. One possible implementation lets the peers query the tracker

for a new list of neighbors after the minimum time interval they have to wait. This interval typically is a parameter of the tracker to prevent peers from flooding the tracker with requests. In each time interval, a peer may learn of new contacts in the swarm and gathers information about them in the same manner as the tracker in the previous approach. Ideally it then knows more contacts than necessary, and can discard a number of non-local neighbors in favor of local ones. However, it can not be assured that a client learns about all existing local peers with this method, since it can not guarantee to receive new information by the tracker with each query.

D. Biased Unchoking

The unchoking mechanism described above is generally unaware of the underlying network topology and leads to a lot of inter-AS-traffic. It only uses an overlay metric $M_O(x)$ in its decision process, which is the download rate in the last 10 seconds in case of BitTorrent and the uploaded blocks in the last 10 seconds in the case of Tribler. Biased unchoking tries to address this problem by including information about the location of the neighbors in the unchoking process. To this end, we introduce an additional metric $M_L(x)$ in the unchoking process which reflects the locality of the neighbor x to the given peer. Examples for $M_L(x)$ are the number of IP- or AS-hops from the peer to its neighbor x. However, $M_L(x)$ can also be more complex and include, e.g., traffic engineering preferences of the ISP.

This locality metric may again be obtained by the peers from a dedicated information server, or by re-using information primarily intended for other services, such as in the Ono plugin. However, we assume that the complexity of computing the locality rating is generally hidden from the peer by the underlay information service it queries for that information. Thus, we only have one value per peer to take into account in the unchoking process.

There are several possibilities to include the locality information $M_L(x)$ in the unchoking process of BitTorrent which we present in the following.

1) Biased Optimistic Unchoking: With regBT, all interested peers that are not regularly unchoked are candidates to be optimistically unchoked. Biased optimistic unchoking reduces this set of candidates to those neighbors which have a good locality value $M_L(x)$, i.e. above or below a certain threshold. Then, a candidate can be chosen from this reduced set. Another option is to unchoke the peer with the best locality value $M_L(x)$.

2) Separate Unchoke Slots: With separate unchoke slots, a peer divides his n unchoke slots into two groups. Then, the first group of m slots is allocated to the neighbors with the best locality values $M_L(x)$. In case several peers have the same locality rating, the overlay metric $M_O(x)$ may be used as a tie-breaker. The remaining $n-m$ slots can be assigned according just to $M_O(x)$. Thus, a peer might reserve 2 unchoke slots for the nearby neighbors regardless of their performance. This might raise fairness issues and incite free-riding by local peers.

3) Combining Locality Information with Overlay Rankings: The idea here is to combine the locality value $M_L(x)$ of a peer with the metric of the overlay $M_O(x)$, e.g., its upload speed, in a way so that the peer can rank its neighbors according to a new combined metric $M_C(x)$. Then, the regular BitTorrent choke algorithm is used with the only modification, that the best neighbors according to $M_C(x)$ get the unchoke slots.

We can combine the two ratings (locality $M_L(x)$ and the overlay metric $M_O(x)$) by weighting and adding them. This results in a new combined metric

$$M_C(x) = \alpha \cdot M_L(x) + (1-\alpha) \cdot M_O(x) \qquad (1)$$

of a neighbor x. It is the weighted sum of the locality value $M_L(x)$ and overlay metric $M_O(x)$. The weight factor α determines to which degree the two metrics impact the result. Then, the ranking amoung the neighbors is done according to the combined metric $M_C(x)$.

IV. Performance Evaluation

The performance evaluation on the different locality-aware peer selection mechanisms was conducted by means of simulation. We will first describe our evaluation methodology including the simulator used.

A. Evaluation Methodology

1) Simulation Model: We conduct steady-state simulations with a stable peer population in one swarm. New peers join the system during the whole simulation time, while peers finishing their download or video go offline and leave the system after an additional seeding time. There are no additional offline times during the lifetime of a peer. In the experiments, we vary the seeding time of a peer, i.e., the time it stays online after receiving the complete file, from 5 to 30 minutes in order to generate different load scenarios. Longer seeding times lead to a higher fraction of seeders in the swarm, thus reducing the load.

Each peer is connected via one AS in a multi-AS network. It has a given up- and downlink capacity exclusively used for the overlay application. We simulate a scenario with 20 stub-ASes and one transit-AS in a star topology, i.e. the stub-ASes are all connected to the transit-AS but not directly interconnected with each other (cf. Figure 1). In the stub-AS we simulate a peer arrival process with exponentially distributed inter-arrival time A and $E[A] = 10s$. This arrival process is equally distributed to the 20 stub AS in order to reduce the number of local neighbors a peer can have.

The peers are connected to the AS with an access speed of 2000 kbit/s downstream and 192 kbit/s upstream which are typical values for the DSL access technology. The initial seed (content provider) is placed in one of the stub-ASes and has an increased upload- and download bandwidth of 1024 kbit/s, respectively. It stays online during the whole experiment. The transit-AS does not contain any peer. We simulate the swarm for x hours and discard the initial warm-up phase of y minutes.

In the case of BitTorrent, the overlay protocol as described in [13] was faithfully implemented, while for Tribler, the

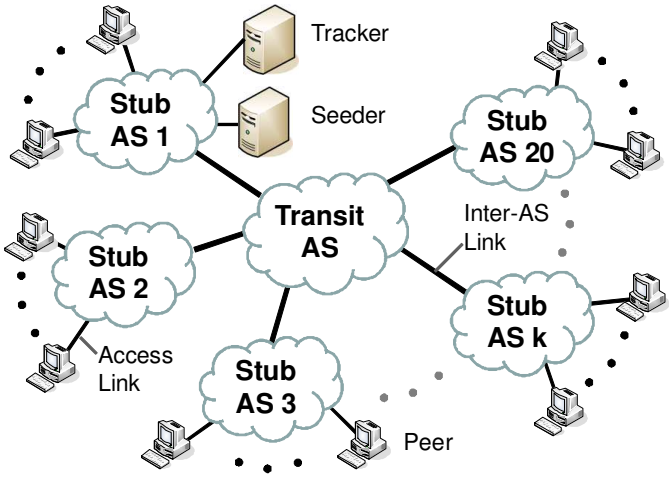

Fig. 1. The simulated topology

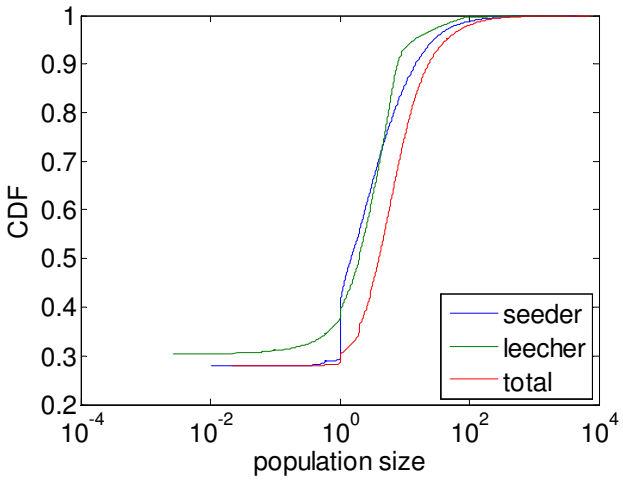

Fig. 2. CDF of swarm sizes.

in grayscales print curves look the same

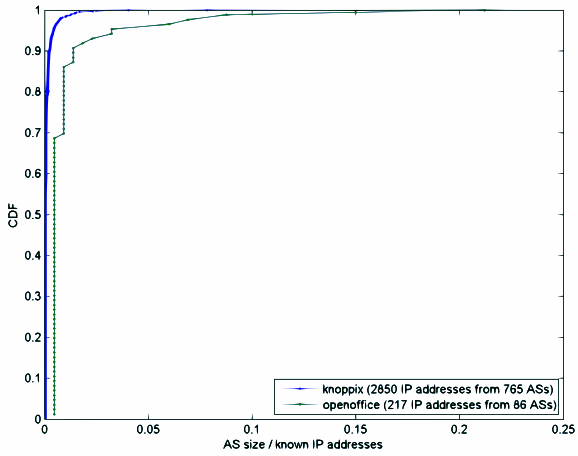

Fig. 3. CDF of the fraction of peers which belong to the same AS.

description in [14] was used for reference as well as the source code.

2) Simulator: Protopeer, flow-based, ...

3) Reasoning for Swarm Model: We present here a study of typical swarm sizes of BitTorrent swarms, in particular the number of lechers and seeds per swarm, which served as a basis for our overlay model. In this study, we measured 63,867 BitTorrent swarms offering video contents, movies, TV series, or documentary. As we sequentially measured the number of seeders and leechers of all swarms, it took about 23 minutes to get data for each swarm. Thus, we obtain one measurement sample per swarm every 23 minutes. The measurements were conducted for a total duration of three days. In order to increase the granularity of our measurement results, especially to describe the dynamics of a swarm, we traced the most popular movies individually. This allows us to obtain the swarm size every 10 seconds for these large swarms.

Fig. 2 shows results from this study. On the x-axis, the x% of the largest swarms are given, i.e. the swarms are sorted according to their population size, while the y-axis shows the cumulated percentage of total peers which belong to the x% of largest swarms. We can see that more than 95% of the observed swarms contain less than 100 peers, so that our simulated swarm is at the larger end of realistic swarm sizes.

Furthermore, we investigate which fraction of a swarm typically belongs to one AS. We measure this for the download of openoffice and knoppix (Fig. 3).

This study shows that for these swarms, the number of peers in one AS is quite small even if the swarm itself is big. The peer population tends to be split up in the network. Therefore, we simulate a swarm being made up of small subgroups of close peers instead of fewer but larger local clusters.

B. Comparison of Biased Unchoking and Biased Neighbor Selection

The focus of our evaluation is on the load scenarios where Biased Unchoking offers an advantage over the standard BT implementation or even over BNS. While Biased Neighbor Selection has been studied intensively [5], [6], [7], [4], to be able to compare it to the BU approach we have to expose the mechanism to the same conditions. We compare the tracker-based Biased Neighbor Selection to the Biased Optimistic Unchoking and a Biased Unchoking with 3 Separate Unchoke Slots reserved for local peers. While a client-based Biased Neighbor Selection offers more flexibility to the peers, we can assure that the maximum number of local neighbors is returned with the tracker-based solution. Thus, this is the best case for the Biased Neighbor Selection.

1) Experiment "BitTorrent": In this experiment, we compare 4 different peer behaviors: (1) regular BitTorrent (regBT), (2) BitTorrent with biased neighbor selection (BNS), (3) BitTorrent with biased unchoking (BU), and (4) BitTorrent with BNS and BU (BNS+BU). In order to assess the performance

from an ISPs side of view, we consider the amount of inter-AS-traffic. To assess the performance from the users point of view, we consider the the download times of the file.

download times comparison will be added? Simon: yes

First, we take a look at the inter-AS traffic. We measured the total bandwidth used for inter-AS connections every second of the simulation. Figure 4 shows the mean of this bandwidth, as well as the 5% and 95% quantiles of this value for the different scenarios and mechanisms.

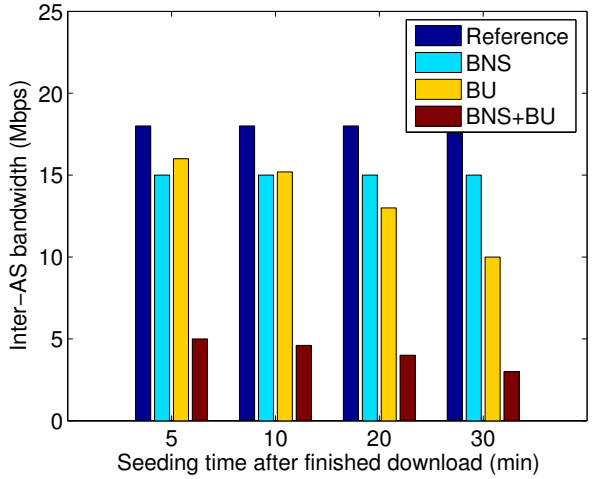

Fig. 4. Medium inter-AS-traffic between the stub-ASes (x-axis: load/mechanism, y-axis:Barplot medium inter-AS bandwidth, 95% quantile, 5% quantile). To be revised later ...

We can observe an effect of the system load on the effectiveness of BU in comparison to the reference implementation and BNS. In scenarios with higher load, the inter-AS bandwidth saving achieved with BU alone is in the same range as BNS, while only little gains can be made in a slightly loaded swarm. The combination of both outperforms each mechanism in every scenario, but also more distinctly in the scenarios with short seeding times. A closer look reveals that the reason for this lies with the preconditions necessary for BU to work effectively.

Tables I shows the mean number of neighbors a peer has during its lifetime, as well as the number of neighbors that express interest in a peer. Table II shows how many local neighbors it knows on average and finally how many local neighbors are interested. While all mechanisms lead to the same medium size of a peer's neighborset and also to the same number of interested peers, they differ in the fraction of these two sets that is made up by local peers. As intended, BNS leads to a higher number of local neighbors.

BU only is effective in the scenarios where there are enough interested peers so that preferring local peers actually has an effect, i.e., more interested peers than the 4 unchoking slots. The number of interested peers grows with a higher load in the system, leading to the observed behavior. This can also be seen in Figure 5, where the CDF of the number of unchoke

	Total Neighbors				Interested Neighbors			
Seeding time	5	10	20	30	5	10	20	30
Reference								
BU								
BNS								
BNS+BU								

TABLE I
TOTAL NEIGHBOR STATISTICS

	Local Neighbors				Local Interested Neighbors			
Seeding time	5	10	20	30	5	10	20	30
Reference								
BU								
BNS								
BNS+BU								

TABLE II
LOCAL NEIGHBOR STATISTICS

slots for local peers is plotted for two load scenarios. We can see that in the highly loaded system, BU and BNS+BU is able to give more unchoking slots to local peers than for a low load, although the number of local neighbors is the same.

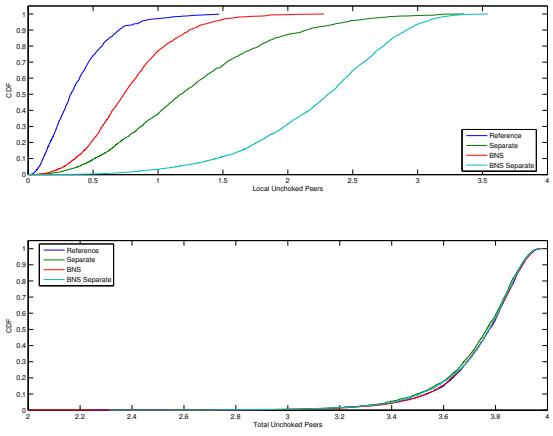

Fig. 5. CDF of unchoke slots given to local interested neighbors (x-axis: number of slots, y-axis:CDF)

Finally, we observe no large impact of the selected mechanisms on the download times of the file (cf. Figure 6). Therefore, we assume that a user will not see any big difference in the performance of the application, while the gains for an ISP are potentially large.

performance of locality mechanisms for changing percentage of local peers? Simon: see below

2) Experiment "Tribler": We conduct the same experiment for the BitTorrent-based VoD overlay Tribler, which mainly differs from BitTorrent in the peer and chunk selection processes. Here, the peer watches the video while downloading it. If frames needed for playout are not available in time, the video stalls until the data has been received. These stall times are used as the primary user performance indicator, since the

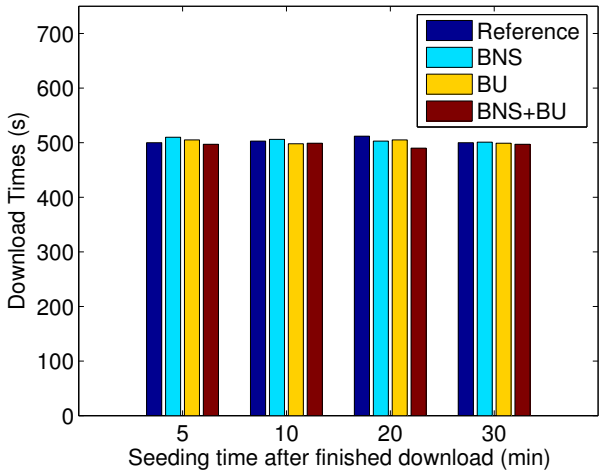

Fig. 6. Medium download times (x-axis: load/mechanism, y-axis:Barplot medium download times, 95% quantile, 5% quantile). To be revised later ...

download time is of no interest compared to the experienced quality of the video.

Apart from these changes, we do not compute the seeding time and consequently the load in the same way as with BitTorrent. Since a Tribler user is watching a video, we let each peer finish watching before adding additional seeding time. Note that while a peer is definitely a seeder by the time it finishes watching, the file may be downloaded much faster than it is watched. Therefore, the seeding time is in general longer than the online time added after a peer stopped watching.

Again, we first consider the inter-AS traffic generated by the swarm. Figure 7 shows the mean of this bandwidth, as well as the 5% and 95% quantiles of this value for the different scenarios and mechanisms.

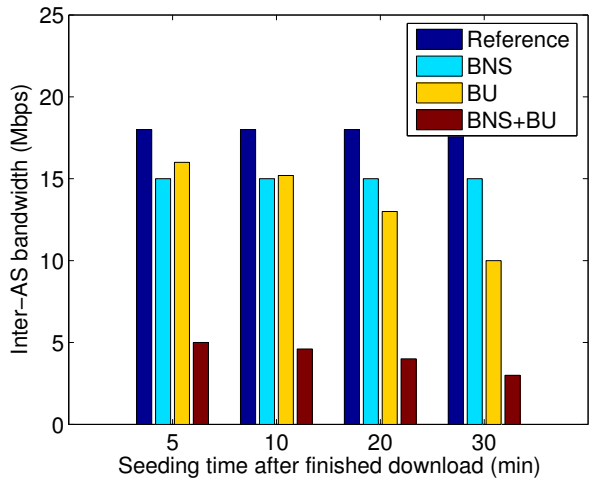

Fig. 7. Medium inter-AS-traffic between the stub-ASes (x-axis: load/mechanism, y-axis:Barplot medium inter-AS bandwidth, 95% quantile, 5% quantile). To be revised later ...

We can observe the same behavior as with BitTorrent. The mechanisms including BU work more effectively in scenarios with higher load. The change in the peer selection shows no effect on the consumed bandwidth, which was to be expected since all peers in the swarm use the same unchoking mechanism and essentially replace the G2G ranking.

Finally, we observe no large impact of the selected mechanisms on the download times of the file (cf. Figure 8). Therefore, we assume that a user will not see any big difference in the performance of the application, while the gains for an ISP are potentially large.

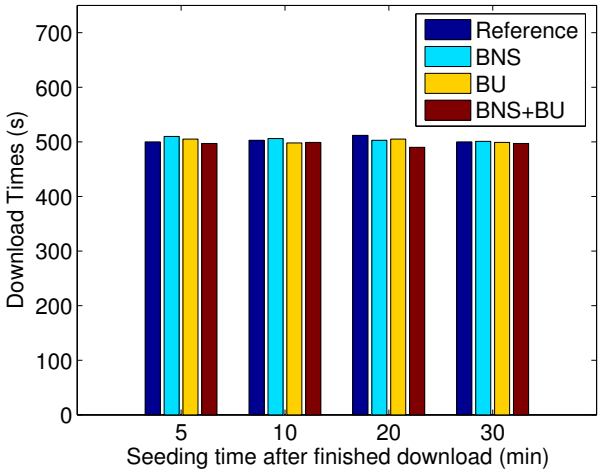

Fig. 8. Medium download times (x-axis: load/mechanism, y-axis:Barplot medium download times, 95% quantile, 5% quantile). To be revised later ...

3) Experiment "Less local neighbors":

Simon: Scenario mit 2, 5 peers pro AS bei gleicher Gesamtpeerzahl -¿ BNS schlechter, BU+BNS sollte gleich gut sein

V. Conclusion

- We'll see...

Acknowledgments

This work has been performed in the framework of the EU ICT Project SmoothIT (FP7-2007-ICT-216259). The authors would like to thank all SmoothIT partners for useful discussions on the subject of the paper.

References

[1] "Bittorrent," http://www.bittorrent.com/.
[2] Bram Cohen, "Bittorrent protocol specification," February 2005.
[3] Ralf Steinmetz and Klaus Wehrle, *P2P Systems and Applications*, Springer Lecture Notes in Computer Science, 2005.
[4] Stevens Le Blond, Arnaud Legout, and Walid Dabbous, "Pushing bittorrent locality to the limit," Tech. Rep., Dec 2008.
[5] Ruchir Bindal, Pei Cao, William Chan, Jan Medval, George Suwala, Tony Bates, and Amy Zhang, "Improving traffic locality in bittorrent via biased neighbor selection," in *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. IEEE, 2006, p. 66, IEEE Computer Society Washington, DC, USA.

[6] Vinay Aggarwal, Anja Feldmann, and Christian Scheideler, "Can isps and p2p systems co-operate for improved performance?," *ACM SIGCOMM Computer Communications Review (CCR)*, vol. 37, no. 3, pp. 29–40, July 2007.

[7] Haiyong Xie, Richard Y. Yang, Arvind Krishnamurthy, Yanbin G. Liu, and Abraham Silberschatz, "P4p: provider portal for applications," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 351–362, 2008.

[8] "Application-layer traffic optimization (alto)," http://www.ietf.org/html.charters/alto-charter.html.

[9] David R. Choffnes and Fabián E. Bustamante, "Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 363–374, 2008.

[10] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki, "Should internet service providers fear peer-assisted content distribution?," in *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, Berkeley, CA, USA, 2005, pp. 63–76, USENIX Association.

[11] Osama Saleh and Mohamed 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*, Washington, DC, USA, 2006, pp. 249–258, IEEE Computer Society.

[12] Hasan Hasan Tobias Hoßfeld Dirk Staehle Zoran Despotovic Wolfgang Kellerer Konstantin Pussep Ionna Papafili George D. Stamoulis Burkhard Stiller Juan Pedro Fernandez-Palacios Gimenez, Maria Angeles Callejo Rodriguez, "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.

[13] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," 2006.

[14] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. Van Steen, and H. J. Sips, "Tribler: A social-based peer-to-peer system," in *In The 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006, pp. 1–6.

TODO LIST