Overlay Connection Usage in BitTorrent Swarms

Simon Oechsner, Frank Lehrieder, and Dirk Staehle

University of Würzburg, Institute of Computer Science, Germany
{oechsner,lehrieder,dstaehle}@informatik.uni-wuerzburg.de

Abstract. The amount of peer-to-peer (P2P) traffic and its inefficient utilization of network resources make it a high priority for traffic management. Due to the distributed nature of P2P overlays, a promising approach for a management scheme is to change the client behavior and its utilization of overlay connections. However, from the viewpoint of a client, there are different categories of overlay connections. In this paper, we discern between these different types of overlay connections in BitTorrent, the currently most popular P2P file-sharing application. A simulation study of BitTorrent and a video-streaming derivate called Tribler provides insights into the usage of these types of connections for data exchange. Thus, traffic management based on client behavior can be optimized by efficiently targeting connections which carry the most traffic. We also show the implications of these results for locality-awareness mechanisms such as Biased Neighbor Selection and Biased Unchoking.

1 Introduction

While the share of traffic generated by peer-to-peer (P2P) file-sharing is decreasing, its absolute amount still increases according to recent studies [CS09]. Moreover, video streaming, which is the projected next top bandwidth consumer, is also partially being conducted via P2P overlays. On the other hand, since P2P traffic is created by end hosts and not by servers, it is much more difficult to control and to manage. Typically, overlays are underlay-agnostic, i.e., they do not consider the underlying network structure in maintaining overlay connections. Therefore, they are comparably wasteful with network resources and therefore also with provider costs. As a result, management schemes for this traffic are currently a major research topic.

The fact that the end user has to willingly participate in such a management scheme is acknowledged since unilateral approaches from providers to throttle P2P traffic have led to a decrease of customer satisfaction. Thus, these schemes have to provide an advantage to end users or at least must not lead to a disadvantage. This is formulated, e.g., in the Economic Traffic Management (ETM) concept of SmoothIT [GRH$^+$08, OSP$^+$08]. Approaches following this guidelines are Biased Neighbor Selection (BNS) [BCC$^+$06] or Biased Unchoking [OLH$^+$09]. Such mechanisms show that influencing decisions at the clients can be an efficient way to manage P2P traffic, in contrast to detecting and re-routing inefficient traffic flows in the core network.

While the most popular file-sharing overlays, such as BitTorrent, form random meshes, not all overlay connections are the same from a specific client's point of view. These different types of connections can be influenced differently by management schemes. Therefore, it is necessary to determine which of these connections carry more traffic, in which direction this traffic flows and which parts of the overlay generate the most traffic.

In this paper, we evaluate the connection usage in BitTorrent swarms, providing insights where traffic management at the clients should be most effective. This information serves as input to ETM discussions such as existing in the IETF working group for Application Layer Traffic Optimization (ALTO). The paper is structured as follows. In the next section, we provide background information on BitTorrent and a video streaming variant, namely Tribler. In the same section, we also review related work. Next, we describe the model for the simulative performance evaluation of these overlays. The results from these evaluations are given in Section 4. We conclude the paper by summarizing our findings in Section 5.

2 Background and Related Work

In this section, we first describe the mechanisms of interest for our evaluation, both of the BitTorrent and the Tribler protocol. These are the neighbor selection and unchoking mechanisms, which are both in the focus of current approaches to ETM. Then we review related work on the topic.

2.1 Key Mechanisms of BitTorrent

The BitTorrent protocol forms a mesh-based overlay and utilizes multi-source download to distribute content. For each shared file, one overlay is formed, a so-called *swarm*. To facilitate the multi-source download, a shared file is split into smaller pieces which are also called *chunks*. These chunks are in turn again separated into sub-pieces or *blocks*. A detailed description of BitTorrent can be found in [LUKM06] and [Bit]. In the following, we will focus on the description of the relevant mechanisms of BitTorrent that are modified for ETM.

Neighbor Set Management. Each peer has only a limited number of other peers it has direct contact with in the swarm. These neighbors know about each other's download progress, i.e., which chunks the other has already downloaded. This enables a peer A to signal its interest in downloading chunks to a neighbor B holding chunks that the local peer still is missing. We say that peer A is *interested* in peer B.

A peer joining a swarm typically initializes its neighbor set by contacting a *tracker*, i.e., an index server with global information about the peer population of a swarm. A standard tracker responds to queries with a random subset of all peers. Peers obtain the address of the tracker for a swarm by downloading a .torrent file from a website. Once a peer A has received a list of contacts in the swarm, it tries to establish connections to them. If it is successful, the according remote peer B is added to A's neighbor set and vice versa.

Thus, each overlay connection has one peer that initiated the connection, and one peer accepting it. In general, new peers fill their neighbor set by initiating most or all of their connections, while peers that have been online longer also have a significant share of connections they accepted. Since 'older' peers have more content and are therefore better sources for chunks, we want to test with our evaluation whether more data flows from peers accepting connections to the initiating ones than in the opposite direction.

Choke Algorithm. Every 10 seconds, a peer decides to which of its interested neighbors it will upload data to. These peers are called *unchoked*, the rest is choked. In standard BitTorrent, there are 3 regular unchoke slots which are awarded to the peers that offer the currently highest upload rate to the local peer. This strategy is called *tit-for-tat* and provides an incentive for peers to contribute upload bandwidth to the swarm. If the local peer has already downloaded the complete file, i.e., it is a *seeder*, the slots are given to all interested neighbors in a round-robin fashion.

Additionally, every 30 seconds a random peer not currently unchoked is selected for *optimistic unchoking* for the next 30 seconds. This allows a peer to discover new mutually beneficial data exchange connections.

Chunk Selection Algorithm. Once a peer is unchoked, it has to select which of the available chunks to download. BitTorrent uses the rarest-first strategy here, i.e., among all chunks the local peer still needs and the remote peer has to offer, the one that is seen the least in the neighbor set of the local peer is selected. This strategy tries to prevent chunks to be shared much less than others, risking the loss of chunks to the swarm.

2.2 Tribler

The VoD client of Tribler adapts the two most important mechanisms of BitTorrent, which are the peer selection strategy in the unchoking process and the piece or chunk selection strategy. We will describe them shortly here, since especially the changed chunk selection mechanism has consequences on the traffic exchange between peers. Details on these mechanisms can be found in [PGW$^+$06].

Chunk Selection Algorithm. The main difference between a file-sharing functionality as offered by BitTorrent and a VoD service as offered by Tribler is that a user of the latter watches the video while downloading it. Thus, the timing for downloading the parts of the complete video file becomes critical, while chunks can be downloaded in any order in a file-sharing network. In particular, chunks in Tribler need to be downloaded roughly in order so that a continuous playback of the video can be ensured.

To this end, the rarest-first chunk selection of BitTorrent is replaced by a strategy based on priority sets. From the current playback position, all chunks until the end of the movie are separated into three sets. The high-priority set contains all chunks with frames from the playback position until 10 seconds after it, while the mid-priority set contains the following 40 seconds of the movie. The remainder comprises the low-priority set. Chunks are first downloaded from the high-priority set, following an in-order strategy within that set. Afterwards, the chunks in the mid-priority set are downloaded, and finally the chunks of the lowest priority, both according to the BitTorrent rarest-first mechanism.

Choke Algorithm. The tit-for-tat (T4T) strategy employed to rate peers in the unchoking process of BitTorrent is based on the assumption that peers can exchange content, i.e., both neighbors forming a connection have some content the other needs. Since the order in which peers download chunks does not matter in the file-sharing application, this is true for enough overlay neighbors. However, with the application VoD peers need to download chunks in roughly the order they are played out as explained above.

Therefore, peers that played back a longer part of the video do generally not need any chunks from peers that are 'behind' them in the playback process. Thus, it is much more probable that data exchange happens only in one direction of an overlay connection in Tribler, namely from a peer that is online longer to a peer that has joined later.

This is taken into account by replacing T4T with a strategy named give-to-get (G2G). G2G favors peers that show a good upload behavior to other peers instead of to the local peer. Consider the local peer A, which has chunks a remote neighbor B wants to download. B then reports to which other peers it has uploaded data in the last δ seconds (by default, $\delta = 10$). Then A queries these peers to make sure that B does not exploit the mechanism. The rating value of B then is the amount of data it forwarded that it originally received from A. Only if there is a tie using this metric, B's total upload is considered as a tie-breaker. This ensures that peers with a high upload capacity are not unchoked by a large number of neighbors, but only by a selected subset.

2.3 Relevance for Economic Traffic Management

ETM approaches utilizing locality-awareness, such as BNS or BU, influence the neighbor selection and the unchoking process of clients. The former method tries to establish connections to close peers, e.g., peers in the same AS. This is mainly achieved by changing the decision to which other peer a local peer will initiate a connection to. If not all peers implement such an ETM mechanism, and if more traffic flows into one direction of a connection than in the other, this leads to an unbalanced effect on the traffic flows of the peers promoting locality, as we show in our results.

On the other hand, BU prefers local neighbors when optimistically unchoking another peer. Thus, it influences the upload of a peer more than its download. Still, due to the T4T algorithm, it has an indirect effect on the download of a peer as well. Another method of relevance here would be the restriction of signaling interest to only a subset of peers instead of all peers holding missing content. However, we will not consider this mechanism here.

2.4 Related Work

A number of measurement and analytical studies on BitTorrent swarms can be found in literature. In [IUKB$^+$04], tracker logs of a 5 month period are evaluated for one swarm. Additionally, a modified client is used to gather data. This work gives some important information about client behavior and swarm dynamics for the BitTorrent protocol current at the time of its publication. While there is no explicit differentiation between overlay connection types of clients, first conclusions are drawn about the composition of a peer's neighbor set. It is stated that a neighbor set contains 'old' as well as 'new' peers, enabling every peer to find some sources for data. This can be mapped to our considerations about initiated connections (to 'older' peers) and accepted ones (from 'newer' peers). Thus, we provide a more in-depth view of this special phenomenon, and additionally evaluate its effect on ETM.

Another trace-based analysis of BitTorrent swarms is provided by [PGES05]. An index site as well as the associated trackers are monitored over a period of 8 months in 2004, with a focus on the availability of system components. This includes the infrastructure that is not part of the core BitTorrent protocol itself. The results cover important aspects like peer uptime, swarm size over time and average download speeds, but take no detailed look at overlay structures.

A more recent and comprehensive measurement study of BitTorrent swarms was conducted in [HHO+09]. Here, data from more than 300,000 swarms active between June 2008 and May 2009 was gathered and evaluated, in addition to existing data sets. Among other things, it was found that swarm characteristics depend strongly on the type of content shared. One of the most important results for this work is the skewness of the peer distribution in the network topology, something which we model in our scenarios.

In [BHP06], a simulative approach to analyze the mechanisms of the BitTorrent protocol is taken. The focus here is on the efficiency of the rarest-first chunk selection and of the tit-for-tat unchoking policy. The study concludes that the load in a swarm is distributed unfairly, since peers with a higher capacity upload more than their fair share. However, this seems to be only a minor problem in active swarms. Nevertheless, mechanisms are presented that limit this unfairness at the cost of protocol efficiency. Additionally, experiments are conducted where nodes join a swarm in the post-flash crowd phase, i.e., when new nodes with little or no content connect to older nodes with many chunks. Similarly, 'pre-seeded' nodes, i.e., nodes having completed most of the file download, are added to a swarm during its initial flash-crowd phase.

In the former case, it is shown that the new nodes quickly download chunks that are interesting to their neighbors, enabling them to participate in the swarm. Here, outgoing connections are considered, but no comparison to incoming connections is made. In the case of pre-seeded nodes joining a swarm, the results show that these nodes might take much longer to complete their download than in a scenario where all nodes have the same share of the content. However, the number of neighbors used in the evaluation scenarios is much lower than in a real swarm with the current version of the protocol, which might influence the results.

Finally, an analytical model for BitTorrent swarms was presented in [QS04]. It captures values like the leecher and seeder population of a swarm and its upload and download capacity to draw conclusions about the evolution of the peer population over time and about the average download time. By necessity, the level of abstraction of the model prohibits taking into account mechanism details such as the neighbor selection or unchoking mechanisms.

None of the works above distinguishes in detail between different connection types in the neighbor set of a peer. To the best of our knowledge, there is no work considering the correlation between traffic flows and the type of connection between two neighbors. Moreover, no such study exists for current approaches to ETM.

3 Simulation Model

Our evaluation is based on a event-driven simulation. In the following, we describe our default simulation scenario and the simulator used. Specific parameters that are changed in the experiments are described in Section 4.

3.1 Simulation Scenario

We simulate one BitTorrent or Tribler 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. In case of Tribler, these values translate to a stream bitrate of slightly below 1 Mbit/s.

We simulate the swarm for 6.5 hours, of which it is in the steady state for 5 hours. New peers join the swarm with an exponentially distributed inter-arrival time A with a mean value of $E[A] = 10\,\mathrm{s}$. As a result, one simulation run consists of about 2300 downloads in the default scenario. The peers stay online for the full download duration of the file plus an additional, exponentially distributed seeding time with a mean value of 10 minutes. Peers do not go offline during the download or the seeding time. As a result, we measured that the swarm contains on average about 120 peers depending on the scenario. These parameters are the same as in [OLH$^+$09].

The peers are connected with an access speed of 16 Mbps downstream and 1 Mbps upstream, which are typical values for the DSL access technology. The seed has a symmetric upload and download bandwidth of 10 Mbps. It goes offline after 1 hour of simulation time. We model the inter-AS links as well dimensioned, i.e., the only bottlenecks are the access links of the peers.

3.2 Simulator

The simulator used in this work is a current version of the simulator used in [OLH$^+$09]. It is based on the P2P simulation and prototyping Java framework ProtoPeer [pro, GADK09]. The simulator contains a flow-based network model adopting the max-min-fair-share principle [BG87]. It faithfully implements the BitTorrent functionality and behavior as described in [LUKM06] and [Bit]. It includes all key mechanisms, in particular the piece selection mechanisms, the management of the neighbor set, and the choke algorithm. Furthermore, the complete message exchange among the peers themselves, between peers and the tracker as well as between the peers and the information service for locality data is simulated in detail. For more specifics, we refer to [OLH$^+$09].

4 Experiments and Results

In this section, we present the results from our simulative performance evaluation. For the measured traffic amounts, we average over all peers in one simulation run, and show the mean values and 95% confidence intervals over 10 runs.

In the following experiments, we vary the seeding time and the interarrival time of the peers and compare the BitTorrent protocol with the Tribler protocol. Additionally, we take a look at the upload and download traffic of an ISP implementing ETM.

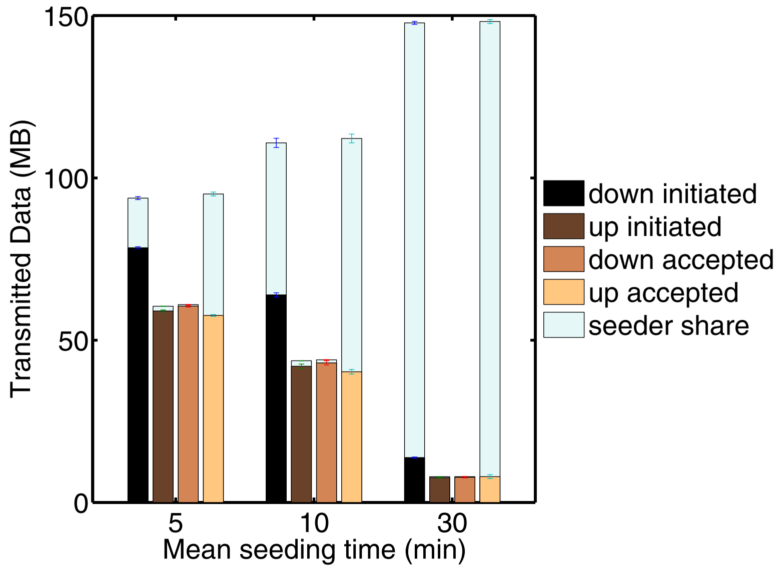

Fig. 1. Traffic direction for different seeding times

4.1 Seeding Time

In the first experiment, we let the peers go offline after mean seeding times of 5, 10 and 30 minutes, cf. Figure 1. Each group of bars shows the average volume of data per peer downloaded and uploaded over initiated and accepted connections, respectively. The light blue bars on top are the share of data that was downloaded from seeders (in case of download traffic), and uploaded while being in seeder mode (in case of upload traffic).

Our first observation is that more traffic is downloaded via initiated connections than via accepted ones, and that similarly, more traffic is uploaded via accepted connections. A large share of this traffic flows from seeders to leechers, although even among leechers, initiated connections are used more heavily for download than accepted ones.

This biased direction of traffic flows is important when trying to influence P2P traffic flows by changing overlay connections, since a mechanism changing the selection of neighbors on the initiator side mainly affects traffic flowing in the opposite direction.

For higher seeding times, the discrepancy between initiated and accepted connections grows. However, this is mainly due to the larger share of traffic uploaded from seeders, which are contacted more often from leechers than the other way around. Thus, depending on the seeding behavior of peers, an ETM changing client behavior should not only work in the leeching mode, but also when the client in question is a seeder.

4.2 Interarrival Time

In this experiment, we test whether the connection usage changes for different swarm sizes and a different peer arrival rate. We compare Poisson arrival processes with a

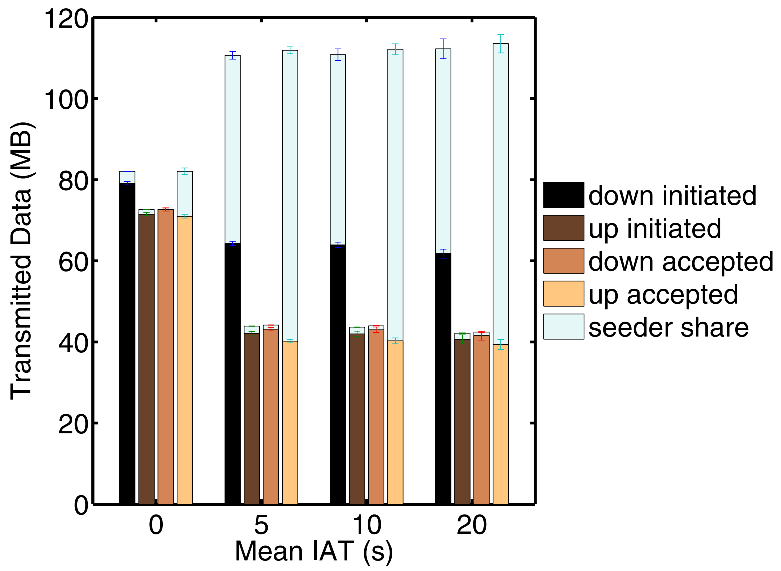

Fig. 2. Traffic direction for different IATs

mean interarrival time of 5, 10 and 20 seconds. Additionally, we evaluate a scenario where 500 peers join the swarm at the same time, i.e., a flash-crowd scenario (denoted by 'Mean IAT 0'). The results are shown in Figure 2.

We observe that, for the swarms in the steady state, the interarrival time and the resulting change in the peer population leads to no significant differences in the connection usage. The effects are the same as explained above. However, this changes for the flash-crowd scenario. Here, the peers all start downloading the file form the initial seed at the same time, and therefore no additional seeders exist in the swarm during most of their leecher phase. Additionally, peers show a similar download progress and establish connections to each other at roughly the same time. As a result, the difference between data flowing over initiated and accepted connections is much smaller than in the steady state scenarios.

4.3 Overlay Type

Next, we compare the predominant BitTorrent protocol with Tribler to see whether the application of VoD streaming changes the distribution of traffic over the different types of overlay connection. To be able to fairly compare the two protocols, we use a fixed deterministic online time of 25 min for all peers. This is long enough to ensure that all peers download or play out the complete file, but is independent from their download mechanisms, in contrast to seeding times.

As discussed in Section 2.2, the in-order download of chunks should lead to more uni-directional traffic flows in the overlay. This can be observed in Figure 3. The

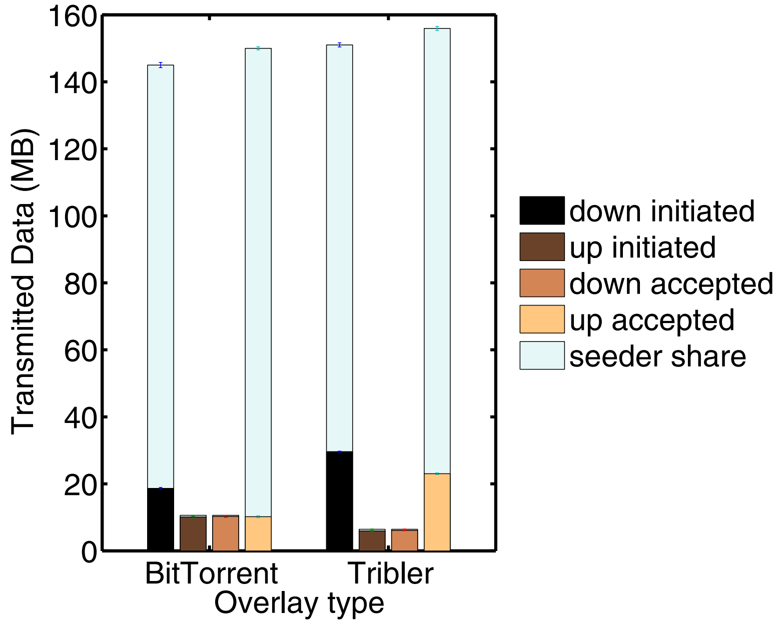

Fig. 3. Traffic direction for different overlay types, BitTorrent and Tribler

difference in traffic flowing over initiated and accepted connections is more pronounced in Tribler, with more traffic flowing to the peers that initiate connections.

Thus, an ETM mechanism that wants to be efficient not only for file-sharing, but also for the future application of video streaming, needs to take into account the inherent hierarchy among peers sharing the same video. A mechanism that governs mainly to which peers connections are initiated influences even more the sources of traffic flowing to the local peer, and even less the direction of upstream traffic flows.

4.4 Effect on Inter-AS Traffic

Until now, we have considered the traffic flows per peer in a regular BitTorrent swarm. To judge the effect ETM mechanisms have, we next take a look at the traffic savings using two different approaches, BNS and BU. In this scenario, we use a swarm setup based on measurements of live BitTorrent swarms [HHO$^+$09]. We use a topology with 20 ASes, and set the probability of a peer going online in AS k is $P(k) = \frac{\frac{1}{k}}{\sum_{i=1}^{20} \frac{1}{i}}$. This models the skewed peer distribution observed in real swarms. All peers use the default BitTorrent protocol, except for the peers in the largest AS 1. These implement the described locality-awareness mechanisms, BNS, BU, and the combination of both. This way, we show the effects on the traffic resulting from a part of the swarm implementing ETM.

We measure both the incoming and the outgoing traffic for each AS. Figure 4 shows the incoming traffic for the four different client implementations in AS1. We observe

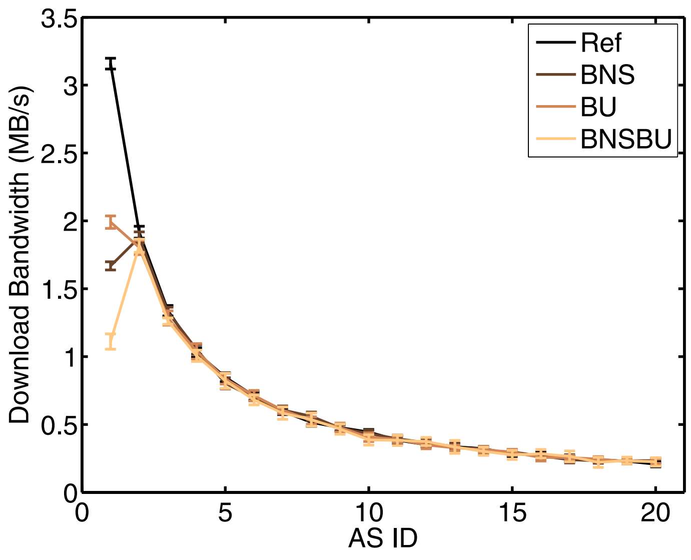

Fig. 4. Download traffic for different ETM mechanisms

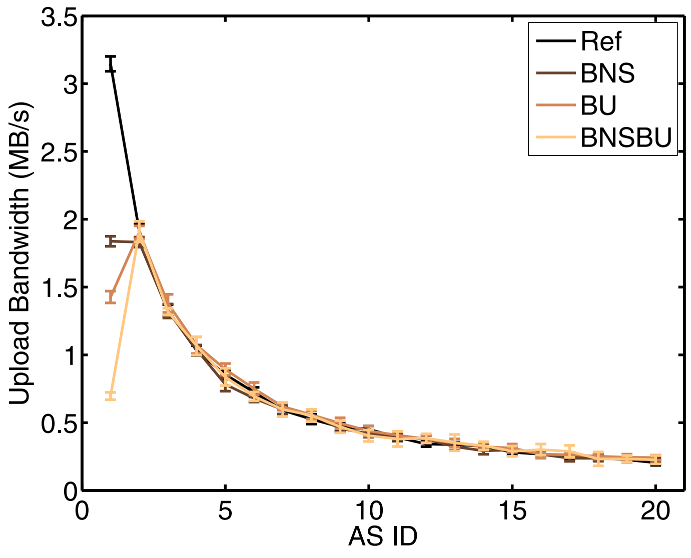

Fig. 5. Upload traffic for different ETM mechanisms

large differences in the effectiveness of the mechanisms, which can be explained by the results above. BNS governs mainly how connections are initiated. In conjunction with the fact that initiated connections carry more incoming traffic, this leads to less consumed download bandwidth in comparison to BU, which influences only the upload of a peer. Still, BU saves traffic due to the tit-for-tat algorithm and because the other peers in AS1 also upload preferredly to local peers.

In contrast, BU proves to be more effective than BNS in reducing the upload of AS1, cf. Figure 5. BNS reduces the upload traffic less, both in comparison to BU and in comparison to the download traffic. This again is due to the biased usage of overlay connections. The combination of both mechanisms reduces the upload traffic much stronger than the download traffic, which can be attributed to the fact that BNS supports BU very well in its preference to upload to local peers, but not the other way round.

5 Conclusions

The results presented in this work show that more traffic flows from peers accepting an overlay connection to the peer initiating it. This effect is enhanced when a peer becomes a seeder, since then it is only an uploader which accepts much more connections than it initiates. Additionally, the amount of traffic flowing from seeders to leechers is significant even for short seeding times. Therefore, we conclude that any efficient ETM has to take this fact into account. Influencing the peer neighbor selection behavior on the initiator's side influences the traffic flows towards the local peer stronger than the opposite direction. Moreover, the ETM mechanism should be as efficient in the seeder mode as in the leecher mode, such as Biased Unchoking using the optimistic unchoke slot, which is utilized in both states. We showed that the described effect is observable in a realistic swarm where only a part of the overlay employs ETM.

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 Tobias Hoßfeld in particular for his help with conducting runs.

References

[BCC$^+$06] Bindal, R., Cao, P., Chan, W., Medval, J., Suwala, G., Bates, T., Zhang, A.: Improving traffic locality in bittorrent via biased neighbor selection. In: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, p. 66. IEEE Computer Society, Washington (2006)

[BG87] Bertsekas, D., Gallagher, R.: Data Networks. Prentice-Hall, Englewood Cliffs (1987)

[BHP06] Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and improving a bittorrent networks performance mechanisms. In: 25th IEEE International Conference on Computer Communications (INFOCOM 2006), pp. 1–12 (April 2006)

[Bit] Bittorrent specification, `http://wiki.theory.org/BitTorrentSpecification`

[CS09] Inc. Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2008-2013. White Paper (June 2009)

[GADK09] Galuba, W., Aberer, K., Despotovic, Z., Kellerer, W.: ProtoPeer: A P2P Toolkit Bridging the Gap Between Simulation and Live Deployment. In: Proceedings of the 2nd International Conference on Simulation Tools and Techniques (2009)

[GRH$^+$08] Gimenez, J.P.F.-P., Rodriguez, M.A.C., Hasan, H., Hoßfeld, T., Staehle, D., Despotovic, Z., Kellerer, W., Pussep, K., Papafili, I., Stamoulis, G.D., Stiller, B.: A new approach for managing traffic of overlay applications of the smoothIT project. In: Hausheer, D., Schönwälder, J. (eds.) AIMS 2008. LNCS, vol. 5127, Springer, Heidelberg (2008)

[HHO$^+$09] Hoßfeld, T., Hock, D., Oechsner, S., Lehrieder, F., Despotovic, Z., Kellerer, W., Michel, M.: Measurement of bittorrent swarms and their as topologies. Technical Report 463, University of Würzburg (November 2009)

[IUKB$^+$04] Izal, M., Urvoy-Keller, G., Biersack, E.W., Felber, P.A., Al, Garcés-Erice, L.: Dissecting bittorrent: Five months in a torrent's lifetim, pp. 1–11 (2004)

[LUKM06] Legout, A., Urvoy-Keller, G., Michiardi, P.: Rarest first and choke algorithms are enough (2006)

[OLH$^+$09] Oechsner, S., Lehrieder, F., Hoßfeld, T., Metzger, F., Pussep, K., Staehle, D.: Pushing the performance of biased neighbor selection through biased unchoking. In: 9th International Conference on Peer-to-Peer Computing, Seattle, USA (September 2009)

[OSP$^+$08] Oechsner, S., Soursos, S., Papafili, I., Hoßfeld, T., Stamoulis, G.D., Stiller, B., Callejo, M.A., Staehle, D.: A Framework of Economic Traffic Management Employing Self-organization Overlay Mechanisms. In: Hummel, K.A., Sterbenz, J.P.G. (eds.) IWSOS 2008. LNCS, vol. 5343, pp. 84–96. Springer, Heidelberg (2008)

[PGES05] Pouwelse, J.A., Garbacki, P., Epema, D.H.J., Sips, H.J.: The bittorrent p2p file-sharing system: Measurements and analysis. In: Castro, M., van Renesse, R. (eds.) IPTPS 2005. LNCS, vol. 3640, pp. 205–216. Springer, Heidelberg (2005)

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

[pro] Protopeer, `http://protopeer.epfl.ch/index.html`

[QS04] Qiu, D., Srikant, R.: Modeling and performance analysis of bittorrent-like peer-to-peer networks. In: SIGCOMM 2004: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 367–378. ACM, New York (2004)