Practicability of T4T Along Cycles

Michael König
Professor: R. P. Wattenhofer
Supervisor: Raphael Eidenbenz

January 11, 2011

Contents

1 Introduction

The popular peer-to-peer file sharing protocol BitTorrent is based on the idea that peers interested in the same "torrent" (a group of one or more files) can trade blocks of data. Such a group of peers is called a "swarm".

It had generally been assumed that selfish behaviour and, more specifically, freeloading (downloading data from a swarm without contributing to it by uploading data) would not pay off in terms of download speed as regular BitTorrent client implementations prefer to send data to peers from whom they also received data. This assumption has been proven false: a free-riding BitTorrent client named *BitThief* achieves regular download speeds without uploading any data [2].

In an effort to disencourage freeloading several *Tit-for-Tat* (T4T) protocols have been proposed with the goal in mind to maximize trading opportunities. *CycleT4T* utilizes cycles of interest that span across multiple swarms (which are not considered by standard BitTorrent). For instance, if a peer A is interested in a peer B which is interested in a peer C which in turn is interested in peer A, these three peers can trade by sending data along the cycle C $\rightarrow$ B $\rightarrow$ A $\rightarrow$ C.

In theory CycleT4T works very well, but it is currently unknown how well it would perform in real swarms and whether it is worth the effort. This semester thesis makes an attempt to answer this question.

2 Experiment Design

2.1 Definition of Cycles

We define a k-cycle as a tuple $(swarm_0, swarm_1, \ldots, swarm_{k-1}, peer_0, peer_1, \ldots, peer_{k-1})$ where $peer_i$ is in both $swarm_i$ and $swarm_{i+1 \mod k}$ and interested in data from $swarm_{i+1 \mod k}$ (i.e. is not a "seeder" in this swarm).

Furthermore, $swarm_0, swarm_1, \ldots, swarm_{k-1}$ as well as $peer_0, peer_1, \ldots, peer_{k-1}$ all need to be different, and we'll treat tuples with the same swarms and peers and merely shifted (not permuted!) indicies as equivalent cycles.

Note that we expect any peer in a swarm to be able to provide interesting data to any peer interested in data of that swarm. This is not too far from reality, especially when employing source coding as suggested in [4].

2.2 Approach

Foremost, our goal is to estimate the number of (non-equivalent) cycles in an existing BitTorrent system and the number of (non-equivalent) cycles any peer is in.

On the one hand, we want to obtain snapshots of BitTorrent swarms made in the shortest possible timespan, and on the other hand, we want to collect the necessary information about peers that allows us to count cycles. Since we need to know whether a peer is a seeder in a specific swarm, we need to actually connect with each peer using the BitTorrent protocol and try to get them to send us their "bitfield" (containing information on what parts of a torrent the peer has obtained).

To conduct these measurements we modified the existing BitThief client such that it won't actually download or upload data but still join swarms, advertise itself and perform Peer Exchange (PEX) in order to acquire and contact as many peer addresses as possible in a "short" amount of time. All measurements were to be done from a single machine.

2.3 Choice of Torrents

2.3.1 Sites

As we wanted to avoid obtaining biased results we used torrents from three different major websites. The Alexa Traffic Rank [1] was used in judging sites. We picked the top three (see Table 1): ThePirateBay.org (TBP), BTJunkie.org (BTJ), KickAssTorrents.com (KAT); all of which seemed pretty easy to crawl and had similar torrent categories.

Website	Alexa Traffic Rank	Total Number of Torrents
thepiratebay.org	94	1,124,616
btjunkie.org	397	564,575
kickasstorrents.com	568	4,012,558
extratorrent.com	762	1,188,202
torrentdownloads.net	763	4,200,519
torrentreactor.net	811	780,678
monova.org	1217	3,236,225
torrenthound.com	1326	2,674,329

Table 1: BitTorrent websites with highest Alexa Traffic Rank [retrieved on 07.28.2010]

We also considered and explored additionally examining private tracker sites, so-called "darknets" (see [5]), but we decided to stick to these few public sites for time limitation reasons.

2.3.2 Data sets

Of all three sites we picked the top 1000 movies as well as the top 1000 TV shows. Both of these categories contain torrents with rather large files. This has two benefits: individual peers are more likely to stay longer in the swarms (thus improving the accuracy of our "snapshot" even if it's done over a larger time window). Moreover, torrents with a large content size and many pieces are more interesting in general. We expected to find more cycles in the TV shows as these contain a lot of TV series which are usually split up into different torrents by season and sometimes even by episode.

In order to examine a more specialized set of torrents we chose to measure the top 1000 "Highres Movies" from TPB as well. It would be interesting to see whether such specialized communities show results that differ from the "mainstream" torrents.

Finally, we also tracked an overall "Top 100 Torrents" list, provided by TPB, containing torrents from all categories.

We also wanted to collect data from the newest 1000 or 5000, but it turned out that these torrents usually barely contained any peers, much less peers that were in multiple of them.

2.4 Time Window Size Choice

Keeping in mind that an average peer stays only for 1 hour in an average torrent, we tried to keep the snapshot time window as small as possible. However, to collect a significant amount of data our first tests showed that we required about 2 hours. Our first tests also showed that the setup we were going to use (see Section 3) could not cope

with more than about 200 of these large swarms at once. In the end we measured our "top 1000" data sets over a time span of 10 hours.

Note that the time window size is not all that critical as, for a large enough data set, the number of peers we missed because they left before we tried to contact them, and the number of peers contacted that joined the swarm within the measurement time window cancel each other out, given that the swarm size stays roughly constant during the measurement.

3 Implementation and Execution

3.1 Implementation

3.1.1 The BitTorrent Client

Using BitThief [2] as a basis we implemented a BitTorrent client with the following special behaviour:

- Our client does not upload or download any data.
- It keeps opening connections to not yet contacted known peers, up to 64 per swarm; accepts incoming connections.
- When a new connection is established (incoming and outgoing alike) it immediately sends a bitfield (claiming to have (randomly chosen) 99% of the pieces) and a PEX packet (in hope to get both back in return).
- Our client closes connections either after the remote peer has sent its bitfield and after it has announced the peers known to it through PEX, or after the connection was open for 60 seconds. (In a few tests we noticed that the bitfield and PEX is usually sent during the first 60 seconds after the handshake if they are sent at all)
- It sets the status of all its connections to *not interested* and *unchoked* and ignores piece request packets.
- It claims to possess 99% of the file when contacting the tracker.

This client furthermore writes a record of the form $(swarm, ip, port, progress, peerid)$ to a local MySQL database each time it closes a connection. The $progress$ field stores how full the received bitfield was, if any was received, and a designated value, if none. Note that if a connection was not accepted on TCP/IP level it is not recorded, but if it was accepted it is recorded even if no BitTorrent handshake took place. The application also creates extensive log files which may be analyzed for further data.

Essentially, our client tries to collect and get to know as many peers as possible and obtain their bitfields, while freeing up connections as soon as possible to be able to connect to more peers.

The limit of 64 connections per swarm was set to ensure fairness between all 100-200 swarms in which the client simultaneously resides. The limit on the total number of connections is imposed by the client implementation and the memory available on the machine used for our measurements.

3.1.2 The Crawler

To collect the .torrent files we implemented a simple crawler, and adjusted it for each of the three websites. For each torrent, it recorded the torrent's name, its category and

seeder/leecher counts as specified by the sites.

3.2 Execution

The measurements took place on a machine in ETH's Open-Net so that our client experienced only minimal firewall and router load restrictions, which removed problems with the amount of connections we previously experienced inside the ETH. The main bottleneck now turned out to be the amount of memory our BitTorrent client implementation used and we could not handle more than 200 swarms with 64 simultaneous connections.

In total 7 useful data sets were recorded, each on a different day and each in at most 10 to 12 consecutive hours. These were:

- TPB Top 100 (976,000 peer records, 100 torrents)
- TPB Movie + TV Shows Top 1000 (1,906,000 peer records, 1000 torrents)
- BTJ Movie Top 1000 (2,306,000 peer records, 820 torrents)
- BTJ TV Show Top 1000 (2,205,000 peer records, 926 torrents)
- KAT Movie Top 1000 (2,882,000 peer records, 954 torrents)
- KAT TV Show Top 1000 (3,028,000 peer records, 992 torrents)
- TPB Highres Movies Top 1000 (2,370,000 peer records, 999 torrents)

(The torrent numbers being lower than the labeled "Top 1000" is caused by broken .torrent files supplied by the sites.)

4 Data Evaluation

With millions of collected data records computing the distribution and the total number of cycles turned out to be difficult. For performance reasons we only consider 2-cycles, 3-cycles and 4-cycles, which should, however, be satisfactory. We also created a few plots and tables unrelated to cycles out of mere curiosity.

For this evaluation of the data we also make the following two assumptions:

- Peers can be uniquely identified by their IP (ignoring their port and peerid).
- Peers which accept connections but don't send their bitfield are leechers. [This assumption is not made for 4.1]

4.1 How many peers did we reach?

To get an idea of the quality and accuracy of the data we collected we compared the number of collected records to the numbers advertised on the torrent sites.

For most individual swarms we reached 3-15% of the peers advertised by the sites, but there were several outliers as low as 0.1% and as high as 30-50%.

Overall statistics for six of the seven datasets can be found in Table 2, visualized in Figure 1. (Note that the "Connections Established" values of "Leechers Only" and "Seeders Only" may not add up to those of "All Peers" values because we didn't know whether someone we connected to was a leecher or a seeder if we didn't receive their bitfield)

The unsually high values for the TPB HR Movies dataset stem from the small swarm sizes and the relatively long measurement window.

For the other datasets the fraction of peers reached seems roughly anti-proportional to the total amount of peers - we reached about 725.000 peers each.

	All Peers			Leechers Only			Seeders Only		
	Adv	BO	CE	Adv	BO	CE	Adv	BO	CE
TPB Videos	1943536	34.9%	63.0%	712122	23.7%	100.4%	1231414	41.4%	85.8%
TPB HR Movies	226038	64.2%	154.7%	131490	56.7%	212.3%	94548	74.7%	291.1%
BTJ Movies	8760816	6.6%	11.6%	3476332	3.6%	16.3%	5284484	8.6%	16.9%
BTJ TV Shows	3498676	10.5%	20.7%	1433682	5.5%	30.4%	2064994	14.0%	31.3%
KAT Movies	2153863	21.6%	39.8%	1205514	11.1%	43.7%	948349	34.9%	76.4%
KAT TV Shows	2265995	14.1%	29.0%	794040	6.9%	49.3%	1471955	18.0%	40.9%

Table 2: Values advertised by sites (Adv), Bitfields Obtained (BO), Connections Established (CE)

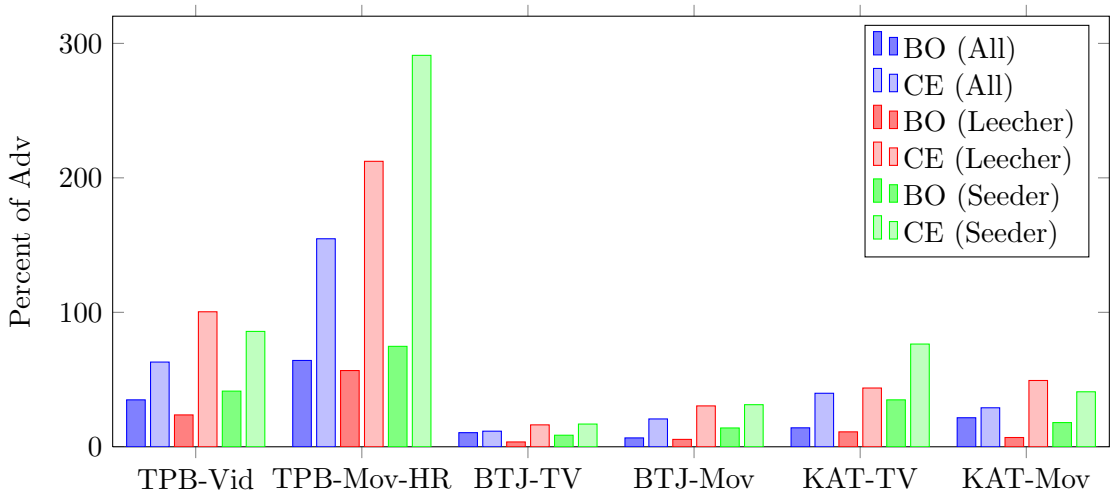

Figure 1: Bitfields Obtained (BO) and Connections Established (CE) compared to values advertised by the sites (Adv)

4.2 In how many swarms is a peer?

#Swarms	1	2	3	4	5	$6-10$	$11-20$	>20
TPB Top 100	77.24%	15.77%	4.36%	1.46%	0.59%	0.52%	0.06%	0.00%
TPB Videos	69.10%	18.48%	6.33%	2.79%	1.36%	1.62%	0.26%	0.05%
TPB HR Movies	57.46%	20.66%	10.63%	5.27%	2.53%	2.98%	0.41%	0.05%
BTJ Movies	63.30%	20.80%	7.87%	3.63%	1.83%	2.23%	0.30%	0.04%
BTJ TV Shows	56.84%	21.96%	9.26%	4.59%	2.54%	3.84%	0.80%	0.18%
KAT Movies	69.01%	19.46%	6.67%	2.55%	1.07%	1.08%	0.14%	0.02%
KAT TV Shows	57.46%	20.98%	9.64%	4.54%	2.52%	3.92%	0.76%	0.18%

Table 3: Number of swarms a peer was observed in

We were also interested in this simple question. Albeit easy to answer, it can help us understand our data better.

Table 3 shows the number of swarms we found peers in. Most notably we see that we have met most peers in only one swarm. Since our measurements have basically only gotten us a small sample of each swarm these numbers likely only present a lower bound - other sources suggest the true fraction of peers only in one swarm to be as little as 15% (see [3]).

We were also curious as to whether participants of larger swarms tend to visit more swarms simultaneously. Figure 2 compares a swarm's size to the average number of swarms we saw the swarm's peers in. However, apparently there is no link to be found here. (The graph basically looked the same for all datasets.)

4.3 Cycles in the Observed Data

Figure 3 shows the total number of 2-cycles, 3-cycles and 4-cycles per swarm (a) and per peer record (b) in our datasets. Several things stand out:

- The ordering is the same for 2-cycles, 3-cycles and 4-cycles.
- TV Shows and HR movies datasets contain the most cycles. This is what we expected: new TV shows often have separate torrents for individual episodes, and most people are either interested in all or in none of the episodes of a TV show; HR movies form a subcommunity of people who prefer high resolution movie torrents over "normal" ones and are thus more likely to meet in their separate set of torrents.
- The datasets with more diverse torrents (TPB top 100 all categories and TPB all video) contain less cycles. This is as expected as well!

Note: The values for 2-cycles and 3-cycles in Figure 3 were computed accurately, the values for 4-cycles are approximated: they are upper bounds with an error margin of at most 1%.

Figure 4 shows the distribution of the cycles among the peers. For performance and computation time reasons these were done with only a subset of the peers, which was 20000 peers in size and randomly chosen from the lists of unique observed peers.

Note: Extrapolating the data for such a set of 20000 peers to all met peers of that dataset actually yielded values close to the total ones from Figure 3 ($< 5\%$ error). As the values for these two figures were computed quite differently this is a strong indicator that no mistakes were made in either calculation!

'CDF' stands for 'cumulative distribution function'. Values on this axis specify a fraction of all peers which are in the number of cycles specified by the corresponding abscissa. For example, in the "TPB Video" dataset about 20% of all peers are part of 10,000 3-cycles.

Note how any peers that we only met in one swarm cannot be part of any cycle, hence the 50-80% peers that are only in one swarm (see Table 3) show up quite visibly in these graphs.

This leads to the interesting observation that peers which are part of multiple swarms instantly are part of about 1,000 - 10,000 3-cycles and 100,000 - 10,000,000 4-cycles.

4.4 Size of Swarm Intersections

We were also curious whether the size of the intersection of peers of two swarms is related to the two swarms' sizes in any way. Figure 5 shows the most significant link I could find. Those are some very particular sorting conditions though and the connection is only really visible for the "largest" 3-5% of swarm pairs... In conclusion, there is barely any connection between swarm size and swarm intersection size.

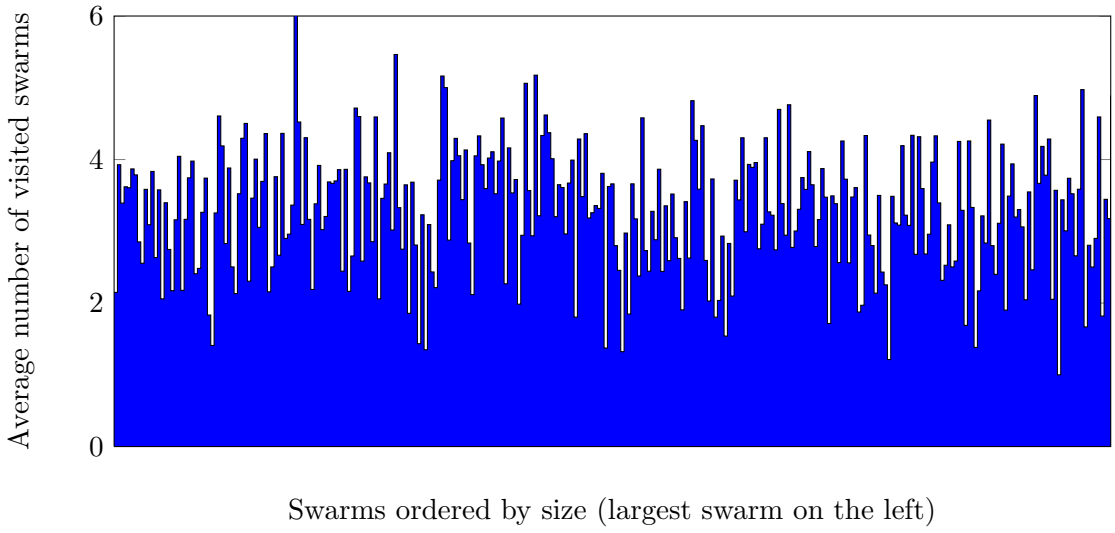

Figure 2: Average number of swarms peers of swarms of certain sizes visit (largest 300 swarms of the BTJ Movies dataset are shown, sizes ranging from 6,800 to 259,700 peers)

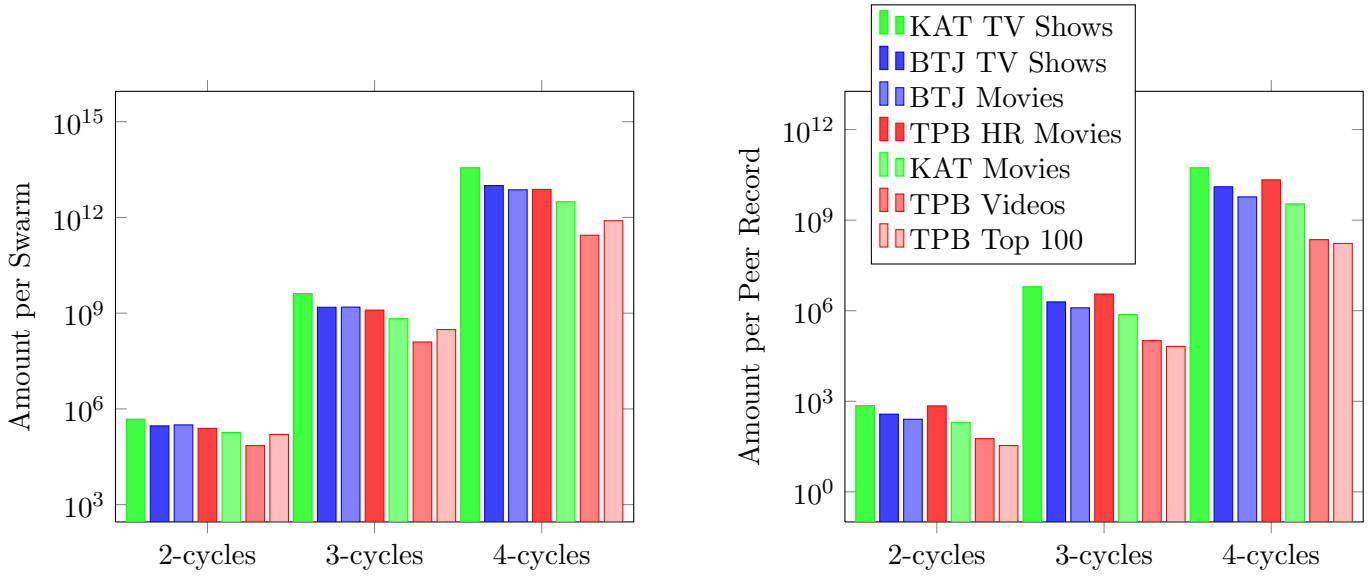

Figure 3: Total number of cycles to be found in our data: (a) per swarm, (b) per peer records

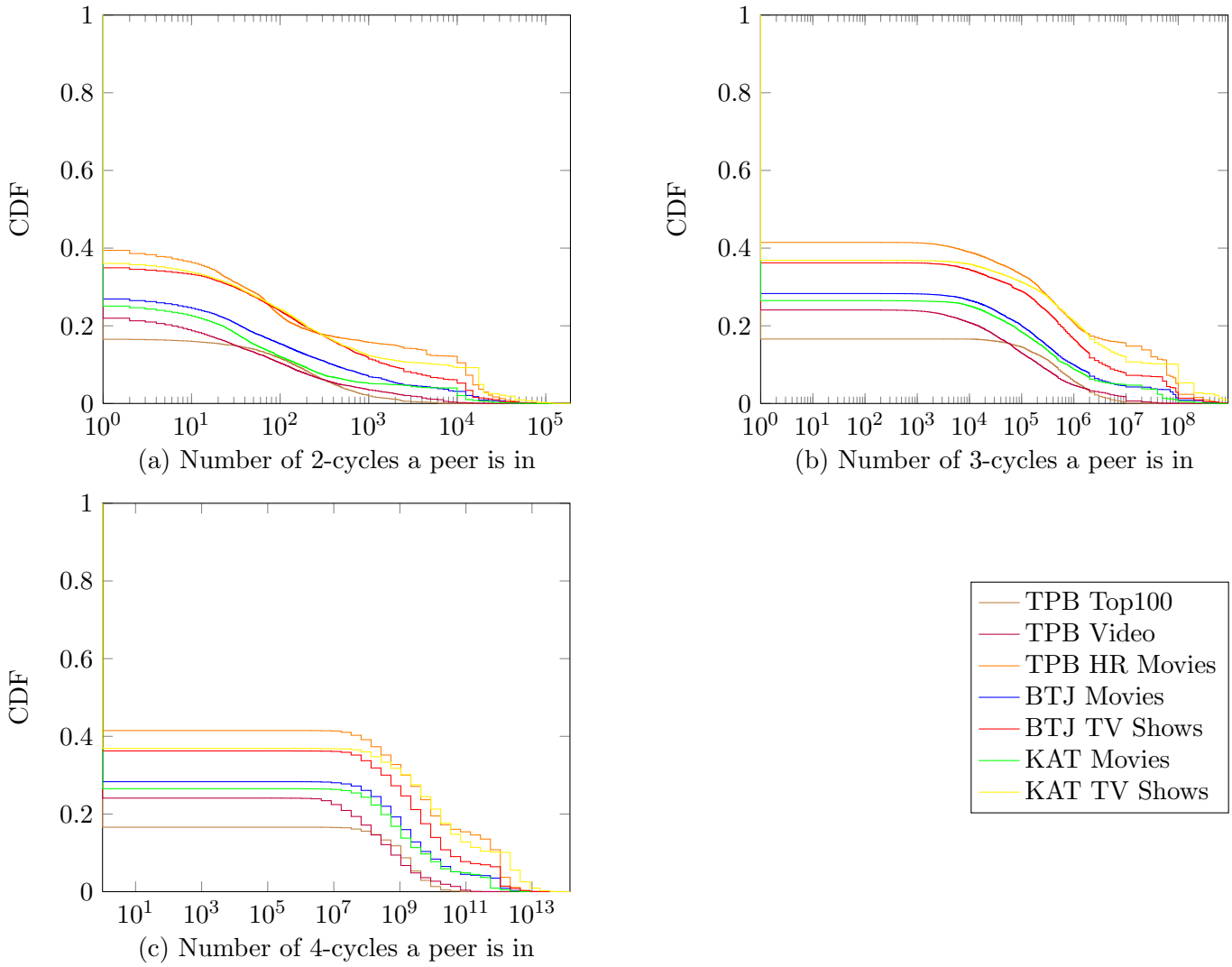

Figure 4: Distribution of observed 2-cycles, 3-cycles and 4-cycles among the observed peers

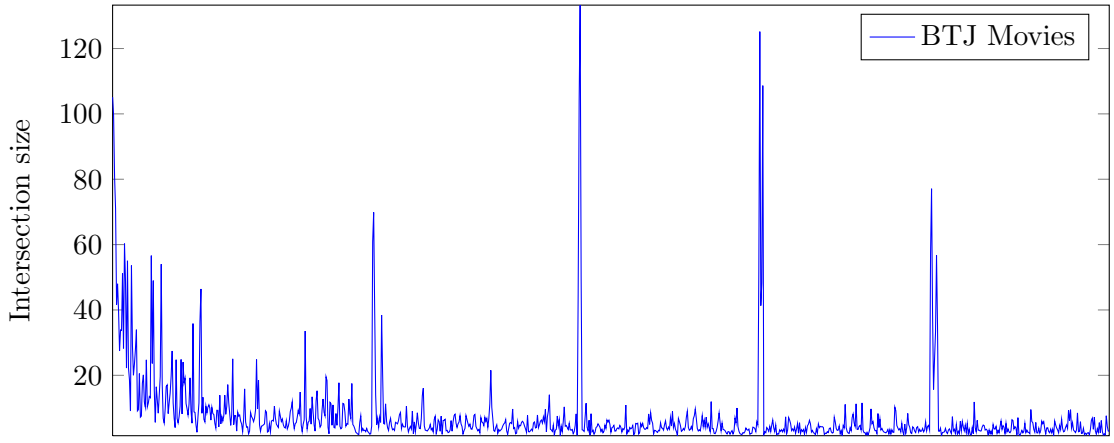

Figure 5: Intersection sizes of swarm pairs. All swarm pairs with non-empty intersection in this dataset were used for this graph (about 140,000). Ordering is by the size of the smaller swarm of the swarm pair, largest on the left with 177,600 peers, smallest on the right with 3,000 peers. To be able to render this graph somewhat properly 1,400 values were averaged and are displayed as one datapoint.

5 Conclusion

We used a specialized BitTorrent client to contact as many peers as possible in thousands of the largest public BitTorrent swarms. The collected data shows that virtually any peer which is part of multiple swarms is part of thousands or even millions 3-cycles and 4-cycles - even though we only use a fraction of all peers to construct these cycles. A couple of assumptions were made during the calculation, and unless these assumptions turn out to be wrong and crucial to the development of these numbers, an extended BitTorrent protocol like CycleT4T appears to be very promising once widespread.

Further work:

- Re-evaluate using ip+port pairs to identify peers
- Re-evaluate under the assumption peers which don't send their bitfield are seeders
- Re-evaluate with both of the above changes
- Collect data from swarms using private trackers ("darknets", see [5]) and comparing it to the public tracker datasets
- Gather better and more accurate data

- by writing an even more specialized BitTorrent client from scratch with lower memory consumption, or
- by using multiple machines to gather data simultaneously

(The common goal being to improve data collection rate and to decrease the snapshot window size)

References

[1] Alexa The Web Information Company, *Site information*, 2010, http://www.alexa.com/siteinfo.

[2] ETH Zürich Distributed Computing Group, *Bitthief - a free riding bittorrent client*, http://www.bitthief.org/, 2010.

[3] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang, *Measurements, Analysis, and Modeling of BitTorrent-like Systems*, pp. 35–48.

[4] Thomas Locher, Stefan Schmid, and Roger Wattenhofer, *Rescuing Tit-for-Tat with Source Coding*, 7th IEEE International Conference on Peer-to-Peer Computing (P2P), Galway, Ireland, September 2007.

[5] Chao Zhang, Prithula Dhungel, Di Wu, Zhengye Liu, and Keith W. Ross, *BitTorrent Darknets*, Proceedings of IEEE INFOCOM 2010, IEEE, March 2010, pp. 1–9.