Computing manuscript No.
(will be inserted by the editor)

Empirical investigation of BitTorrent community graphs

Tamás Vinkó · Bea Botyánszki

Received: date / Accepted: date

Abstract Most of the users of a BitTorrent community participate in multiple swarms, usually simultaneously as uploader and downloader. Thus the inter-swarm aspect of the bandwidth resource allocation has high relevance in the algorithm design of deployed BitTorrent clients. This inter-swarm connections among the users of a BitTorrent community can be represented by a flow network of special structure. Using this representation it has been demonstrated that *de facto* solutions to the resource allocation in BitTorrent communities are suboptimal. In this paper we investigate this conclusion in more details using graph measures and optimization techniques. We find that BitTorrent communities are sensitive for removal of highly contributing users, that the inter-swarm connectivity can lead to different performance results, and that torrent selection mechanisms can hardly improve the average download performance of users. Regarding the theoretical optimum of the discussed problem we show that it does not necessary comply with the BitTorrent protocol.

Keywords BitTorrent · private communities · resource allocation · flow networks · network science

1 Introduction

Resource allocation problems in distributed systems can be defined in various ways [1]. Focusing on the allocation of bandwidth, one particular goal is to maximize the total throughput of the system. Although simple solutions to this problem does not usually lead to fair allocations [2], it maximizes the average downloading speed of the users in the system. This paper deals with this particular problem in the context of

T. Vinkó, B. Botyánszki
University of Szeged, Institute of Informatics
H-6720 Szeged, Árpád tér 2, Hungary
Tel.: +36-62-546 193
Fax: +36-62-546 397
E-mail: tvinko@inf.u-szeged.hu

the most widely used distributed content sharing system, BitTorrent [3]. At its highest level, BitTorrent is a collection of *swarms*. A swarm consists three components: content, seeders and leechers. The *content* in a swarm is a file or collection of files to be shared by the participating users, who are the *seeders* and *leechers*. Seeders are those users who have and share the complete copy of the content. Leechers are the other type of users, those who are actually downloading the content. The *size* of a swarm is the number of leechers and seeders in that swarm. One of the smartest ideas behind BitTorrent is that the content is split into small pieces and during the downloading phase, leechers exchange pieces among themselves. Thus leechers can also be uploaders. This idea makes the BitTorrent protocol very efficient (in terms of downloading speed) and highly scalable (the larger the swarm size the more efficient the network is).

After finishing the download of the requested content, the BitTorrent protocol, by definition, does not require to stay in the swarm as active seeder. This situation can easily lead to dead swarms, i.e. seedless swarms. One possible and quite popular solution for this problem is setting up a BitTorrent *community* [4,6]. In these communities a server, also called a *tracker*, requests the users to register themselves and it maintains a database about the user's activity. Though this solution brakes the distributive aspect of the BitTorrent ecosystem, it usually leads to a very attractive and efficient content sharing systems. This is due to the so-called *sharing ratio enforcement* in which the users are obliged to seed up to a prescribed upload-to-download ratio[1], usually around 0.7. This means that after downloading 1GB content, the user must upload at least 0.7GB.

This paper puts forward the investigation of the *inter-swarm* connections within a community done in the papers [8,9]. The inter-swarm resource allocation is concerned with the fact that many BitTorrent users are taking part in multiple torrents at the same time, usually both as leecher and seeder. Thus, the BitTorrent clients have built-in mechanisms to make decisions on which torrents they should be seeding in and how the bandwidth should be divided among the swarms they are participating in. As it was demonstrated in [9], the currently applied solutions can perform as low as 50% compared to the optimal. With further experimental analysis we show (i) how sensitive a BitTorrent community could be to removal of users, (ii) what is the role of inter-swarm seeding, and (iii) if it is possible to improve the efficiency with different torrent selections. We will also investigate the theoretical upper bound and show that it does not necessary fulfil the requirements of the BitTorrent protocol.

2 Models and Datasets

2.1 Graph models

This paper attempts to investigate some part of the conclusions of Capotă *et al.* [9], namely the inter-swarm bandwidth allocation problems in BitTorrent communities regarding the maximum throughput. For that reason here we use the same flow network

[1] Other seeding incentive mechanisms are existing and successfully used in communities, e.g. based on effort [7]

Table 1 Model parameters describing a BitTorrent community

Notation	Definition
I	set of users in the community
T	set of torrents in the community
S_t, L_t	set of seeders and leechers participating in torrent $t \in T$
μ_i, δ_i	upload and download bandwidth of user i
Λ_i	library of user i

model as it was already defined and used in [9]. For the sake of the completeness, the most important definitions are repeated here.

We use the notations summarized in Table 1. The inter-swarm connections in a BitTorrent community at a certain time instant[2] is represented by the flow network $G = (V, E, f, c)$, where V is the set of vertices, E is the set of edges, f is the flow function, and c is the capacity function. This is a directed bipartite graph, where V is the disjoint union of three subsets of vertices, U, L and D, and with each edge in E connecting two vertices that are in distinct subsets. These three sets of vertices are defined in the following way:

– the upload nodes $U = \{u_1, \ldots, u_m\}$ represent the upload potential of the m active users (both seeders and leechers);
– the leeching sessions nodes $L = \cup_{i \in I} L_i$ represent the participation of leechers in torrents, where $L_i = \{l_i^t \mid i \in L_t, t \in T\}$ is the set of leeching sessions of user i; and
– the download nodes $D = \{d_1, \ldots, d_n\}$ represent the download potential of the leechers.

We can see that a user i is represented by the set of nodes $\{u_i, d_i\} \cup L_i$. The set of edges E represents the transfer potential from the upload nodes to the download nodes through leeching session nodes. If user i is leeching in a torrent t, then there are *leeching edges* from u_i to the leeching sessions of all other users in torrent t. Similarly, if a user i is seeding in a torrent t then there are *seeding edges* from u_i to the leeching sessions of all other users in torrent t. The set of seeding edges is denoted by E_s. All of the edges thus defined are called *upload edges*. Finally, to represent downloading, the graph also has edges from the leeching session nodes to download nodes. These edges are called *download edges*.

The capacity and flow functions of G are defined as follows:

– the *capacity function* $c : U \cup L \cup D \to \mathbb{N}$ represents the bandwidth of peers, where $c(u_i) := \mu_i$ is the upload bandwidth of user i, $c(l_i^t) := \infty$, and $c(d_i) := \delta_i$ is the download bandwidth of user i, and
– the *flow function* $f : E \to \mathbb{R}^+$ represents the bandwidth allocation, having the property of flow conservation: for all $l_j^t \in L$ the equality $\sum_{u_i \in U} f(u_i, l_j^t) = f(l_j^t, d_j)$ holds.

It is easy to see that any flow in G is equivalent to a particular bandwidth allocation in the corresponding BitTorrent community G represents.

[2] For the shake of simplicity we avoid the notion of time in the graph models.

2.2 Network flow problems

Using the flow network model of BitTorrent communities as it was described in the previous subsection now we define the two problems to be considered.

Maxflow Maximizing the throughput in a BitTorrent community is equivalent to solve the maximum flow problem [17] in the community's flow network representation. As it was used in the paper [9] the corresponding linear programming formalism is the following

$$\begin{aligned} & \max \textstyle\sum_{(u_i,l_j^t)\in E} f(u_i,l_j^t), \\ \text{s.t.} \quad & \textstyle\sum_{t,j} f(u_i,l_j^t) \le \mu_i \qquad \forall u_i \in U, \\ & \textstyle\sum_t f(l_j^t,d_j) \le \delta_j \qquad \forall d_j \in D. \end{aligned}$$

For solving the Maxflow problem on the actual graph instances we used the `maxflow` function of the `R/igraph` package, which is an implementation of the Goldberg–Tarjan algorithm [18].

BTflow At any given point in time in a BitTorrent community, the total throughput produced by the online users is called *BTflow*. We consider this throughput value as a result of a (suboptimal) flow network algorithm. In this paper we use the simulator written by Mihai Capotă (Technical University of Delft, the Netherlands), which was also used and validated in the paper [9]. We note here that although the simulator is also providing us with the details about the total flow, i.e. what is the flow value it puts to the edges, we do not intend to use this information in this paper. Our aim is to see what kind of conclusions one can have using only the total flow value of the BTflow. This means that our experiments in the following sections can be done with any simulator or method which produces reasonable approximation of BTflow values.

2.3 BitSoup, FileList

The investigated graphs are taken from the same datasets used in [9]. These datasets are measurement traces of two private BitTorrent communities: *BitSoup* and *FileList*. The most important properties of the traces are summarized in Table 2. It can be clearly seen that the two communities differ in all aspects. As it was done in [9], one additional dataset is used here, which was derived from the BitSoup trace in a way that the seeding capacities got reduced of all users to produce a dataset with an overall ratio of two leeching sessions per seeding session. This dataset, which will be referred as *Synthetic* in the rest of the paper, represents characteristics of open BitTorrent communities.

From these datasets the graph instances were created, representing the status of the community with respect to the upload and download potentials. The basic properties of these graphs are listed in Table 3. We will be using 10 instances of BitSoup and Synthetic, and 8 instances of FileList. The upload and download capacities were randomly selected from the measurement data of [21].

Table 2 Properties of the datasets [9], with 95% confidence intervals for means.

	BitSoup [19]	FileList [20]
Registered users	84 007	91 745
Total torrents	13 741	3 236
Mean active torrents	6 869.6 ± 30.8	512.2 ± 10.2
Mean active sessions	76 370.3 ± 1 135	32 829.4 ± 672.8
Mean seeders/leechers ratio	5.125 ± 0.155	3.65 ± 0.2

Table 3 Basic properties of flow networks: number of nodes and edges, size of the set U.

	BitSoup			FileList			Synthetic		
graph	nodes	edges	$\lvert U \rvert$	nodes	edges	$\lvert U \rvert$	nodes	edges	$\lvert U \rvert$
1	37 266	1 254 129	16 958	23 705	1 963 645	12 886	29 039	478 089	9 713
2	41 243	1 526 799	18 128	25 516	5 673 353	14 364	32 965	606 040	10 845
3	33 651	934 852	14 622	27 715	4 341 702	14 193	26 928	382 925	8 807
4	33 261	739 941	15 193	26 178	3 331 577	14 236	25 644	274 148	8 543
5	34 075	1 184 615	15 154	26 731	2 997 262	15 291	26 791	537 086	8 812
6	36 849	775 404	16 613	26 215	6 168 009	13 959	28 872	27 2791	9 628
7	33 493	611 714	15 588	24 719	3 562 633	13 365	25 471	198 556	8 576
8	29 426	682 401	13 418	27 796	4 671 512	14 466	22 672	312 499	7 577
9	30 242	508 129	14 084				22 718	187 612	7 644
10	37 504	927 018	16 434				29 747	472 057	9 863

3 Empirical results

3.1 Network robustness

In the first experiment we are interested to see what kind of degree distribution our graphs follow. A graph's degree distribution function $P(k)$ shows what is the proportion of nodes with degree k. For a large set of real-world networks it has been shown (e.g. [22]) that they are not random (i.e. their degree distribution does not follow a Bernoulli distribution function), rather their degree distributions are power-law type, of form $P(k) \sim k^{-\alpha}$.

In order to see whether our graphs have power law degree distribution we used the `power.law.fit` function of `R/igraph`. More precisely, we investigated only the degree of the U nodes. The results are shown in Table 4. For the three communities the obtained α value, the $x_{\min}$ value (that is the index of the node from which the graph follows power law degree distribution), and the p value of the Kolmogorov-Smirnov goodness test (if the p value is greater than 0.05 then the graph is power law) are shown.

The obtained α values are relatively large. In case of $2 \leq \alpha \leq 3$ a power law graph is also called scale free [23]. We got larger values here, so our graphs should not be called scale free. Interestingly, large α value has been shown in networks of mobile phone calls [24]. In these kind of networks, in which the distribution is decaying so quickly, the usual characteristics of scale free networks (such as hubs) are not present. We can conclude, though, that the BitTorrent community graphs are of long-tail.

Table 4 Power law degree distribution of graphs.

	BitSoup			FileList			Synthetic		
graph	α	$x_{\min}$	KS.p	α	$x_{\min}$	KS.p	α	$x_{\min}$	KS.p
1	6.82	395	0.276	6.76	736	0.932	6.59	402	0.999
2	4.64	263	0.013	22.33	2190	0.999	5.48	287	0.518
3	4.85	227	0.207	10.07	1554	0.976	5.41	226	0.186
4	6.63	300	0.998	8.28	1111	0.493	8.89	335	0.999
5	10.62	628	0.993	8.28	1114	0.999	13.80	531	0.868
6	5.21	218	0.995	15.88	2231	0.846	5.54	147	0.783
7	5.30	183	0.934	12.15	1489	0.340	6.37	148	0.994
8	9.46	504	0.951	5.29	1116	0.001	21.37	550	1.000
9	5.04	193	0.785				7.00	177	0.963
10	5.84	352	0.464				8.19	352	0.626

Based on the results above, we now check the effect of nodes removal. Especially, we are interested to see the variation of the BTflow/Maxflow ratio in case of deleting nodes representing users from the networks. The following iterative procedure was executed. In each step 100 users are removed using the one of the following two rules: (i) select random users (*random*), or (ii) select the users with the largest degree (*kMax*).

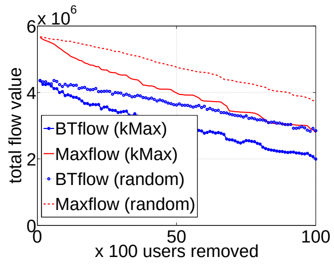
The results of these experiments are shown on Figure 1. Note that removing a user which is leeching at least in one swarm results in removal of more than one nodes from the flow network representation. It can be clearly seen, especially in the FileList experiment, that removing random users leads to smoother decay, contrasted to the *kMax* type removal, in the flow values. In general, the *kMax* removal results in larger drops in the flow values. Note that while the Maxflow value monotonically decreasing (not strictly, though), the BTflow values sometimes got slightly improved. We can also notice that the BTflow/Maxflow ratio does not change radically.

Implications. These experiments proves our expectations based on the degree distribution experiments above, that the BitTorrent community graphs are sensitive for removal of large degree users. These indicate that a community should pay attention on keeping the heavy contributors on-line, since they play essential role in the total throughput[3]. Although private BitTorrent communities usually apply promotion mechanisms to encourage their users for higher contribution, these mechanisms are mostly focused on the amount of data [5]. Our results clearly show that high number of active connections is also need to be incentivized.

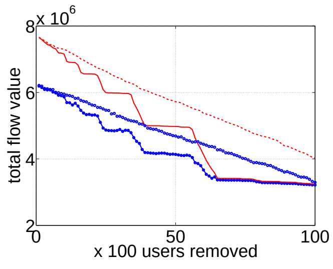
3.2 One swarm per seeder

In the following experiment we are interested to see what is the role of seeders' interswarming. Thus we split the flow networks in such a way that each and every U nodes representing a seeder is connected only to one swarm. Due to the structure of the flow

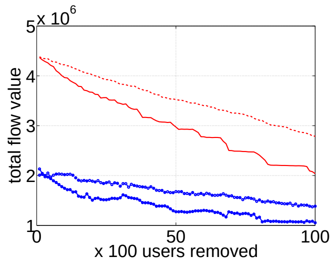
[3] Note that in these particular experiments we did not take the users' bandwidths *directly* into account



(a) BitSoup



(b) FileList



(c) Synthetic

Fig. 1 Removing users.

network model, this does not mean that the seeder nodes would have degree equal to 1, the actual degree depends on the number of leechers in the seeded swarm. Also, this splitting does not necessary lead to a flow network which consists of independent clusters of swarms, because it can happen that the swarms are connected through leechers.

The following procedure was done for a flow network: for each seeding node s all the seeding edges but one were removed from the library Λ_s. We applied two possible rules for selecting the torrent to be kept in the library: (i) select randomly from Λ_s (*random*), or (ii) select one from the torrents in Λ_s which has the most leechers (*maxleecher*). Note that this simple procedure can result in graphs with seedless torrents. Thus post-processing is needed which can be done in the following way.

Step 1 Collect all the seedless torrents into the set T_0.
Step 2 For all elements t of T_0 do the followings: Let s_o be the original seeder of torrent t. Check whether the torrent currently seeded by s_o had multiple seeders. If not, then select another seeder s_o from the original seeder of t. Otherwise, let s_o be the seeder of torrent t.

The procedure was run for every graphs of the three communities, leading to the following results. On Figure 2 we can see the BTflow/Maxflow ratio of the original graphs together with the ratios on the modified flow networks using the two removal rules discussed above. We obtained three different results here. For the BitSoup graphs both cases led to drops in the performance. For the FileList graphs the *random* rule resulted in performance drops, but the *maxleecher* rule sometimes led to slightly better ratio. For the Synthetic graphs we obtain significantly better performance using any of the two rules. Since the Synthetic graphs are derived from the BitSoup graphs, the results we obtained here using the *maxleecher* rule are the same for the corresponding BitSoup graphs, and roughly the same using the *random* rule.

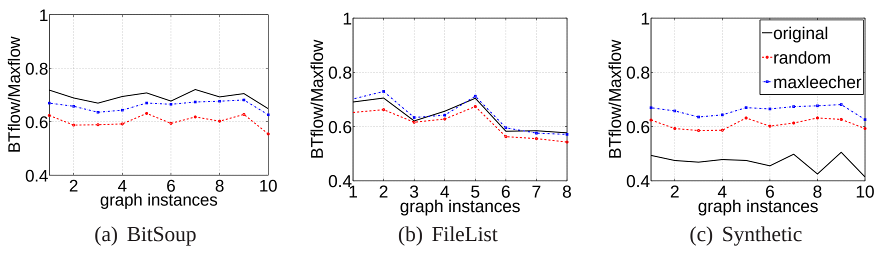


Fig. 2 Results of the one seeder per swarm experiments

In order to see what is the difference between the flow values in the modified networks with respect to the original ones we refer to Figure 3. While the values got decreased in BitSoup and FileList, it is interesting to see the opposite case for the Synthetic graphs.

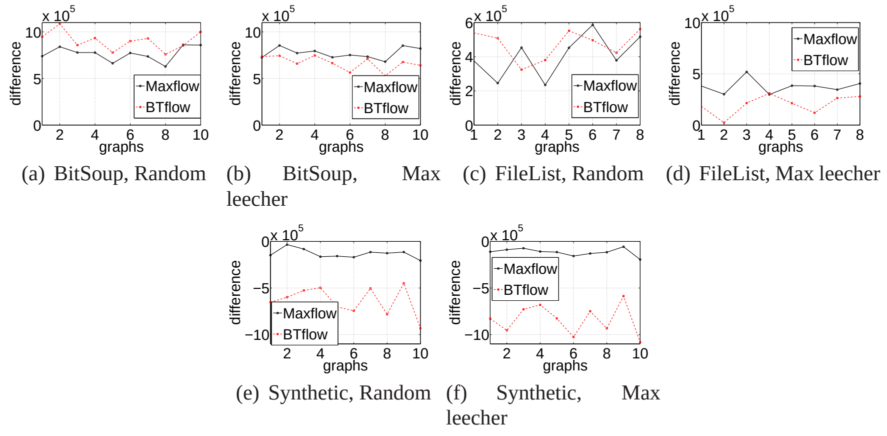


Fig. 3 Absolute differences in the one seeder per swarm experiments

Implications. The results are promising in the sense that for the BitSoup graphs (which are representing closed BitTorrent communities in which the participating users must follow sharing ratio enforcement, thus it is usual that peers are seeding in multiple torrents at the same time) it is worse for the community if the seeders are not following inter-swarm uploading. On the other hand, in open BitTorrent communities (represented by the Synthetic graphs), where no explicit rules are (and could be) applied regarding the seeding, it is better if each seeder uploads in one swarm only. According to Kash *et al.* [25], observation of a private BitTorrent community revealed that many peers are seeding in multiple torrents. While it is practically impossible to find relevant measurements regarding the open BitTorrent, Cuevas *et al.* [26] shows that, on average, a regular publisher (user who injects new content into the BitTorrent network) seeds concurrently only in one torrent.

3.3 Optimization on growing networks

Motivated by the fact we have noticed in Section 3.1 that removing user nodes and the corresponding edges from the flow network, it is possible that the BTflow value gets improved. In this experiment a greedy-type optimization algorithm is proposed which is working on a growing network and systematically tries to identify particular edges which removal leads to local improvement in the BTflow value while keeping the Maxflow value at the same level.

Algorithm 1 Greedy optimizer algorithm

```
Require: Flow network $G$.
 1: $k := 0$; $G_k :=$ empty flow network;   $H_k :=$ empty flow network;   $T_S := \emptyset$;   $T_N := T$;
 2: while $k < |T|$ do
 3:     $k := k+1$;
 4:     repeat
 5:         $t :=$ Uniform($T_N$);
 6:         $U_k := \{u_j \mid (u_j, l_i^t) \in E(G)\}$;
 7:         $N_k := U_k \cup \{l_i^t \mid i \in I\} \cup \{d_i \mid (l_i^t, d_i) \in E(G)\}$;
 8:         $Q_k := \{(u_j, l_i^t) \mid u_j \in N_k\} \cup \{(l_i^t, d_i) \mid d_i \in N_k\}$;
 9:         $V(G_k) := V(G_{k-1}) \cup N_k$;   $E(G_k) := E(G_{k-1}) \cup Q_k$;
10:         $V(H_k) := V(H_{k-1}) \cup N_k$;   $E(H_k) := E(H_{k-1}) \cup Q_k$;
11:         if $k = 1$ then break;
12:     until $U_k \cap V(H_{k-1}) \neq \emptyset$
13:     $T_S := T_S \cup \{t\}$;   $T_N := T_N \setminus \{t\}$;
14:     $b_k :=$ BTflow($H_k$);
15:     $m_k :=$ Maxflow($G_k$);
16:     if $b_k < m_k$ then
17:         $M_k := N_k \cap V(G_{k-1})$;
18:         $\mathscr{P}_k :=$ PowerSet($\{(u_i, l_j^{t'}) \in E_s(G) \mid u_i \in M_k, l_j^{t'} \in M_k\}) \setminus \emptyset$;
19:         $H'_k := H_k$;
20:         for all $P_k \in \mathscr{P}_k$ do
21:             $E(H_{temp}) := E(H_k) \setminus P_k$;
22:             $V(H_{temp}) := V(H_k)$;
23:             if Maxflow($H_{temp}$) $= m_k$ and BTflow($H_{temp}$) $> b_k$ then
24:                 $H'_k := H_{temp}$;
25:                 $b_k :=$ BTflow($H'_k$);
26:             end if
27:         end for
28:         $H_k := H'_k$;
29:     end if
30: end while
31: return $H^* := H_{|T|}$;
```

In the following, we refer to the lines of Algorithm 1 for the formal description. The input is a flow network G. The algorithm iterates through the set of torrents T. The variables T_S and T_N are the sets of selected and non-selected torrents, respectively (line 1). In lines 4–12 the algorithm tries to find a torrent which grows the optimized network in such a way that it remains connected (i.e. the number of its clusters equals to 1). In order to do so, it selects a torrent uniformly at random from T (line 5). Then, in lines 7–10 the flow network G_k is constructed that it contains G_{k-1} together with all leeching session nodes representing the torrent selected in line 5, plus all those nodes which are connected to these leeching session nodes. Note that N_k is the set of these nodes (line 7). The flow network H_k, to be locally optimized, is constructed similarly (line 10). In case the intersection of the new upload nodes and the set of nodes in H_{k-1} (assuming that $k > 1$) is empty, then the algorithm needs to select another torrent. We need to ensure that torrent t is not selected in the next iterations (line 13). If the ratio of the BTflow value of H_k and the Maxflow value of G_k equals to 1, then H_k cannot be improved. Otherwise, select those nodes from N_k which are also in $V(G_{k-1})$ and put them into the set M_k (line 17). These nodes are connected to multiple torrents (i.e. they are connected to leeching nodes representing multiple torrents). For all *seeding* nodes in M_k compose the set $\mathscr{P}_k$ which is the power set of all seeding upload edges (line 18). For all $P_k \in \mathscr{P}_k$ identify which edge set P_k should be deleted in order to improve the ratio r_k (lines 21–25). In this loop, the algorithm systematically tries to identify those seeding edges which can be deleted without decreasing the Maxflow value and, at the same time, leading to increase of the BTflow value. In order to do so, the procedures for calculating the Maxflow and BTflow have to be executed, thus this loop is the most expensive part of the optimizer method.

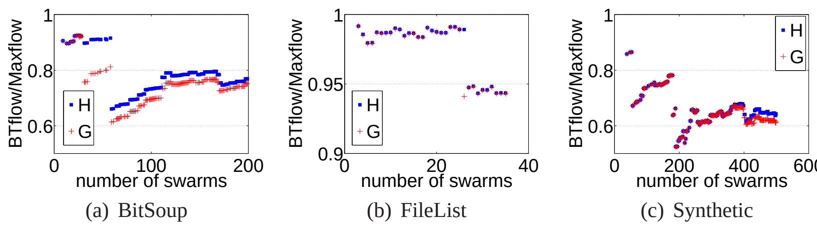


Fig. 4 Results of the greedy optimizer on the growing flow networks

The results of the greedy algorithm for the communities are shown on Figure 4. For all three cases the BTflow/Maxflow values are shown for the original growing graph (G) and for the optimized one (H). Note that due to the high resource requirement of the algorithm implementation (in particular, we had memory allocation issues for the FileList experiment) we could run the algorithm up to a limited number of torrents.

Implications. It can be clearly seen that the proposed optimization method can improve the BTflow/Maxflow ratio only for graphs with very few swarms. That is the

Table 5 Results after deleting seed edges with zero flow

	BitSoup			FileList			Synthetic		
graph	0f	ratio	0f ratio	0f	ratio	0f ratio	0f	ratio	0f ratio
1	0.689	0.718	0.637	0.653	0.690	0.672	0.271	0.494	0.492
2	0.687	0.689	0.610	0.588	0.705	0.719	0.287	0.475	0.464
3	0.649	0.669	0.633	0.633	0.621	0.653	0.242	0.469	0.467
4	0.676	0.694	0.649	0.650	0.657	0.634	0.243	0.478	0.473
5	0.612	0.709	0.653	0.659	0.704	0.701	0.221	0.475	0.464
6	0.688	0.677	0.625	0.455	0.583	0.599	0.249	0.455	0.437
7	0.697	0.720	0.637	0.427	0.585	0.586	0.242	0.498	0.481
8	0.591	0.693	0.629	0.423	0.577	0.565	0.193	0.425	0.428
9	0.654	0.705	0.669				0.224	0.505	0.516
10	0.517	0.649	0.604				0.149	0.414	0.441

case for BitSoup and Synthetic. However, as the number of swarms are increasing, the difference between the performance in the optimized graph and the original graph tends to vanish. Thus, we can conclude that the performance of the BTflow algorithm cannot really be improved with the torrent selection mechanism our greedy algorithm finds.

4 On the upper bound

Now we turn our attention to the theoretical optimum of the maximal throughput in the flow networks representing statuses of BitTorrent communities. Up until now, we have been considering only the *value* of the maximum flow. In this section we also take into account the details of the solution, i.e. where the flow values are taken, what is the flow value put on the edges. The results of this section *could* depend on the actual algorithm solving the maximum flow problem. We use here the `maxflow` procedure from the `R/igraph` package.

4.1 Seed edges with zero flow

In the first experiment we run the maximum flow algorithm and using the results we remove those edges from the networks on which the actual flow value equals to zero. Using these modified graphs, we check what is the total flow value produced by the BTflow algorithm. This edge removal rule is basically a solution to the torrent selection mechanism.

The results are shown in Table 5. For each communities we show the ratio of the seeding edges among all the edges of the network (column 0f), the BTflow/Maxflow ratio in the original graph (’ratio’), and the BTflow/Maxflow ratio in the new graphs (’0f ratio’). First of all, we notice that the ratio of the seeding edges with zero flow can be as high as 69%. However, we conclude that BitTorrent’s performance is usually dropped using this rule, only some cases it got slightly better.

Table 6 Results after deleting upload edges with zero flow.

	BitSoup			FileList			Synthetic		
graph	0f	ratio	0f ratio	0f	ratio	0f ratio	0f	ratio	0f ratio
1	0.693	0.718	0.793	0.653	0.690	0.839	0.279	0.493	0.599
2	0.691	0.689	0.802	0.588	0.705	0.936	0.295	0.474	0.600
3	0.654	0.669	0.795	0.633	0.620	0.829	0.253	0.468	0.625
4	0.682	0.694	0.821	0.651	0.656	0.907	0.257	0.478	0.631
5	0.616	0.708	0.794	0.659	0.704	0.907	0.228	0.475	0.601
6	0.694	0.677	0.766	0.455	0.582	0.731	0.265	0.455	0.642
7	0.704	0.720	0.800	0.427	0.584	0.879	0.259	0.498	0.703
8	0.596	0.692	0.851	0.423	0.576	0.697	0.203	0.425	0.633
9	0.662	0.704	0.827				0.241	0.505	0.705
10	0.522	0.648	0.729				0.158	0.414	0.648

4.2 Upload edges with zero flow

In the following we extend our rules used in the previous subsection to *all* uploading edges at which the maximum flow algorithm puts zero flow value. The results of these experiments are shown on Table 6. It can be clearly seen the increase of BitTorrent's performance. However, according to the BitTorrent tit-for-tat mechanism, a leecher must be uploading to other leechers in the same swarm. Thus we must emphasize that these solutions are infeasible.

Nevertheless, the results of this experiment shows that if we relax the requirement of sharing the content for *some* users then a community can obtain significantly better (8–23%) performance increment regarding the total throughput.

4.3 The Maxflow$_\ell$ problem

What we have seen in the previous subsections is the fact that the BitTorrent performance could be improved by deleting the uploading edges with zero flow values (which are set to be zero by a maximum flow algorithm), but this is violating the rules of BitTorrent. In the following we extend the optimization model of the maximum flow problem defined in Section 2.2. In this extended model, using an additional variable $\ell \geq 0$, it is required for each and every leeching edges to put as large flow value as possible while keeping the sum of the total flow values at the possible maximum level. Note that practically it is possible to set up 0 kB/s upload bandwidth in a BitTorrent client (if a user wants to do so), but in this case the client would be downloading contents only from seeders. On the other hand, considering a case in which there is only one leecher in a swarm seeded by some other users, then this leecher does not need to be uploading. In the flow network model used in this paper this leecher is *not* included among the *u* nodes.

The *Maxflow*$_\ell$ linear programming (LP) model can be formalized in the following way:

$$\begin{array}{lll} & \max \ell + \sum_{(u_i,l_j^t)\in E} f(u_i,l_j^t), & \\ \text{s. t.} & \sum_{t,j} f(u_i,l_j^t) \le \mu_i & \forall u_i \in U, \\ & \sum_t f(l_j^t,d_j) \le \delta_j & \forall d_j \in D, \\ & f(u_i,l_i^t) \ge \ell \ge 0 & \forall (u_i,l_i^t) \in E_\ell, \end{array}$$

where $E_\ell \subset E$ is the set of those upload edges which belong to u_i nodes representing leechers in the flow network.

Solving this LP model for all the graphs we obtained $\ell = 0$. This means that the value of the maximum flow (as in the original model) can be kept only in case we put zero flow values for some leecher uploading edges.

Another LP model can be constructed in which one defines the variable ℓ specific for each leeching edges. This model formulated as

$$\begin{array}{lll} & \max \sum_{(u_i,l_j^t)\in E_\ell} \ell(u_i,l_j^t) + \sum_{(u_i,l_j^t)\in E} f(u_i,l_j^t), & \\ \text{s. t.} & \sum_{t,j} f(u_i,l_j^t) \le \mu_i & \forall u_i \in U, \\ & \sum_t f(l_j^t,d_j) \le \delta_j & \forall d_j \in D, \\ & f(u_i,l_i^t) \ge \ell(u_i,l_i^t) \ge 0 & \forall (u_i,l_i^t) \in E_\ell. \end{array}$$

Solving this second model for the graphs of the three communities we got the following results shown in Table 6. Firstly, the value of the total throughput is slightly decreased (by only 1% compared to the original maximum flow value, so this decrease is negligible). Secondly, we obtained positive ℓ values for *some* leeching upload edges, but there are many of those having zero flow values. The ratio of these leeching edges with zero flow is at least 95%, but usually even higher. We know that this is unacceptable solution in the BitTorrent protocol.

Implications. Our experiments in this section put the earlier obtained results about the BitTorrent performance regarding the maximal throughput into different perspective. Earlier we have seen that using BTflow a community gets to about 50–70% of the optimal performance. However, the theoretical upper bound does not necessary fulfil the requirements of BitTorrent (at least using the classical maximum flow algorithm). Thus we conclude that the BTflow –at least in the form we have been using it in this paper– might not be as far away from the optimal performance as it is suggested by earlier results. Possible improvements for real world systems include implementation of a distributed maximum flow (DMF) algorithm into the BitTorrent client. There are two main problems to be considered: (i) the DMF must respect the BitTorrent-specific constraints discussed above, and (ii) mixed environment of traditional and this specialized BitTorrent clients would not necessary guarantee optimal performance.

5 Related work

Recently the analysis of the seeding bandwidth allocation at inter-swarm level has attracted the researchers attention. In Meng *et al.* [10] analysis of the amount of time required for all peers to get the files in peer-to-peer networks is given. Based on a fluid-model the theoretical lower bound is derived and an optimal algorithm (of exponential running time), together with heuristics are given. For multi-swarm multi-party P2P conferencing systems Liang *et al.* [11] propose optimal cross-swarm bandwidth sharing strategies to address the bandwidth challenge. However, these papers did not compare their solutions to the ones applied in BitTorrent clients. The papers [12, 13] study the bandwidth allocation mechanisms of BitTorrent-like systems using synthetic traces only and finding out that indeed BitTorrent performs sub-optimally. In our paper we use measurement traces of real communities. Regarding the flow network model, Zhong *et al.* [14] evaluate the topological characteristics of overlay BitTorrent networks using a graph model which is different from the one we use. Delaviz *et al.* [16] use several network (graph) measures to understand the behavioural aspects of a BitTorrent reputation mechanism. Finally, the recent paper of Hu *et al.* [15] investigate the inter-swarm aspect among BitTorrent communities, thus it is at one level higher than our focus.

6 Summary and Conclusions

Using a flow network model we investigated a particular resource allocation problem in BitTorrent communities, in which the aim was to maximize the total throughput in the system. In the actual experiments a simulator was used in order to calculate to throughput which would have been provided by standard BitTorrent clients. Based on measurement traces of two communities we discovered that the underlying flow network representation has long-tail degree distribution and thus it is robust against removal of average users. On the other hand, due to the same reason, high degree nodes (users who contribute in many torrents, regardless of their bandwidth) play essential role in keeping the high performance. The torrent selection mechanism can be simulated with deleting edges from the flow network representation. Our experiments with the extreme edge removal, which led to representation of one-swarm-per-seeder scenarios, revealed that depending on the original structure of the graph this can lead to both performance drop (in closed communities) or increase (in open communities). As it was already claimed in [9], the actual torrent selection mechanism employed in many BitTorrent clients is efficient enough, provided that it is coupled with efficient bandwidth allocation mechanism. Our experiments done here with a greedy optimization technique on growing flow networks confirms this finding. On the other hand, closer look on the solutions given by a maximum flow algorithm, which provides the theoretical upper bound on the bandwidth allocation problem we considered, disclosed that they represent skewed torrent allocations putting zero flow values to most of the edges. This leads to the conclusion that the performance of the standard BitTorrent clients are not as low as it was asserted.

Acknowledgements This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013). T. Vinkó was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

References

1. A. Wierzbicki, Trust and Fairness in Open, Distributed Systems, Vol. 298, Springer, 2010.
2. T. Lan, D. Kao, M. Chiang, A. Sabharwal, An axiomatic theory of fairness in network resource allocation, in: Proceedings of the IEEE INFOCOM, 2010, pp. 1–9.
3. B. Cohen, Incentives build robustness in BitTorrent, in: Workshop on Economics of Peer-to-Peer systems, Vol. 6, 2003, pp. 68–72.
4. X. Chen, Y. Jiang, X. Chu, Measurements, analysis and modeling of private trackers, in: IEEE Tenth International Conference on Peer-to-Peer Computing (P2P), 2010, pp. 1–10.
5. X. Chu, X. Chen, A. Jia, J. Pouwelse, D. Epema, Dissecting darknets: measurement and performance analysis. ACM Trans Internet Technol 13:125, 2014
6. Z. Liu, P. Dhungel, D. Wu, C. Zhang, K. Ross, Understanding and improving ratio incentives in private communities, in: IEEE 30th International Conference on Distributed Computing Systems (ICDCS), 2010, pp. 610–621.
7. R. Rahman, M. Meulpolder, D. Hales, J. Pouwelse, D. Epema, H. Sips, Improving efficiency and fairness in p2p systems with effort-based incentives, in: IEEE International Conference on Communications (ICC), 2010, pp. 1–5.
8. T. Vinkó, F. Santos, N. Andrade, M. Capotă, On swarm-level resource allocation in BitTorrent communities, Optimization Letters 7 (2013) 923–932.
9. M. Capotă, N. Andrade, T. Vinkó, F. Santos, J. Pouwelse, D. Epema, Inter-swarm resource allocation in BitTorrent communities, in: IEEE International Conference on Peer-to-Peer Computing (P2P), 2011, pp. 300–309.
10. X. Meng, P.-S. Tsang, K.-S. Lui, Analysis of distribution time of multiple files in a P2P network, Computer Networks 57 (15) (2013) 2900 – 2915.
11. C. Liang, M. Zhao, Y. Liu, Optimal bandwidth sharing in multiswarm multiparty p2p videoconferencing systems., IEEE/ACM Trans. Netw. 19 (6) (2011) 1704–1716.
12. R. S. Peterson, E. G. Sirer, Antfarm: Efficient content distribution with managed swarms, in: NSDI, 2009.
13. R. J. Dunn, S. D. Gribble, H. M. Levy, The importance of history in a media delivery system, in: IPTPS, 2007.
14. L. Zhong, X. Wang, M. Kihl, Topological model and analysis of the P2P BitTorrent protocol, in: 9th World Congress on Intelligent Control and Automation (WCICA), 2011, pp. 753–758.
15. C. Hu, D. Shan, Y. Cheng, T. Qin, Inter-swarm content distribution among private bittorrent networks., IEEE Journal on Selected Areas in Communications 31 (9-Supplement) (2013) 132–141.
16. R. Delaviz, N. Zeilemaker, J. A. Pouwelse, D. H. J. Epema, A network science perspective of a distributed reputation mechanism, in: IFIP Networking, 2013, pp. 1–9.
17. L. R. Ford, D. R. Fulkerson, Maximal flow through a network, Canadian Journal of Mathematics 8 (1956) 399–404.
18. A. V. Goldberg, R. E. Tarjan, A new approach to the maximum-flow problem, J. ACM 35 (4) (1988) 921–940.
19. N. Andrade, E. Santos-Neto, F. Brasileiro, M. Ripeanu, Resource demand and supply in BitTorrent content-sharing communities, Computer Networks 53 (4) (2009) 515–527.
20. J. Roozenburg, Secure decentralized swarm discovery in Tribler, Master's thesis, Delft University of Technology (2006).
21. T. Isdal, M. Piatek, A. Krishnamurthy, T. Anderson, Leveraging bittorrent for end host measurements, in: S. Uhlig, K. Papagiannaki, O. Bonaventure (Eds.), Passive and Active Network Measurement, Vol. 4427 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 32–41.
22. L. A. N. Amaral, A. Scala, M. Barthelemy, H. E. Stanley, Classes of small-world networks, Proceedings of the National Academy of Sciences 97 (21) (2000) 11149–11152.
23. G. Caldarelli, Scale-free networks: complex webs in nature and technology, Oxford University Press, 2007.

24. J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, A.-L. Barabási, Structure and tie strengths in mobile communication networks, Proceedings of the National Academy of Sciences 104 (18) (2007) 7332–7336.
25. I. A. Kash, J. K. Lai, H. Zhang, A. Zohar, Economics of BitTorrent communities, in: Proceedings of the 21st international conference on World Wide Web, ACM, 2012, pp. 221–230.
26. R. Cuevas, M. Kryczka, A. Cuevas, S. Kaune, C. Guerrero, R. Rejaie, Is content publishing in BitTorrent altruistic or profit-driven?, in: Proceedings of the 6th International Conference, Co-NEXT ’10, ACM, 2010, pp. 1–12.