Optim Lett (2013) 7:923–932
DOI 10.1007/s11590-012-0477-5

ORIGINAL PAPER

On swarm-level resource allocation in BitTorrent communities

Tamás Vinkó · Flávio Santos · Nazareno Andrade · Mihai Capotă

Received: 15 May 2011 / Accepted: 27 March 2012 / Published online: 10 April 2012

Abstract BitTorrent is a peer-to-peer computer network protocol for sharing content in an efficient and scalable way. Modeling and analysis of the popular private BitTorrent communities has become an active area of research. In these communities users are strongly incentivized to contribute their resources, i.e., to share their files. In BitTorrent terminology, users who have finished downloading files and stay online to share these files with others in the network are called seeders. The combination of seeders and downloaders of a file is called a swarm. In this paper we examine and evaluate the efficiency of the resource allocation of seeders in multiple swarms. This is formulated as an integer linear fractional programming problem. The evaluation is done on traces representing two existing BitTorrent communities. We find that in communities, particularly with low users-to-files ratio (which is typically the case), there is room for improvement.

Keywords BitTorrent · File sharing system · Resource allocation · Integer optimization

1 Introduction

BitTorrent has enabled the emergence of communities that work as powerful content distribution systems [1,2]. In these communities, users provide the resources to

T. Vinkó (✉) · M. Capotă
Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands
e-mail: t.vinko@tudelft.nl

F. Santos
Universidade Federal de Rio Grande do Sul, Porto Alegre, Brazil

N. Andrade
Universidade Federal de Campina Grande, Campina Grande, Brazil

distribute files to their peers through *swarms*. Peers within a swarm are divided into two classes. Those peers who have and share the complete copy of a file are called *seeders*, while peers who are downloading the file are called *leechers*. Note that a peer in a community can be both seeder and leecher simultaneously in multiple swarms. Because asymmetric Internet connections prevail among domestic users, the upload bandwidth available in a swarm is frequently a bottleneck for the download speed. Seeders alleviate this problem and are therefore paramount for download performance.

In this paper, we evaluate how well seeder resource allocation (to be defined later in Sect. 2) serves downloaders in two BitTorrent communities. Each seeder currently allocates its resources among previously downloaded files autonomously. It is not clear whether this strategy yields desirable results in practice. We devise here an algorithm which serves as a tool for investigating the margins for further optimization.

Considerable research and development effort has been invested in designing and evaluating incentive mechanisms to promote seeding (e.g. [6,8,10]), but the analysis of the allocation mechanisms at inter-swarm level has received less attention. The paper [9] deals with the problem of channel-resource imbalance in multi-channel peer-to-peer systems, which is similar to the problem considered in this paper, however, their provided solution is heuristic based. The approaches proposed in the papers [3,7] are similar to ours, but the scenarios are studied only under synthetic workloads.

We take the complementary approach of analyzing traces from two BitTorrent communities to contrast normally used resource allocation mechanisms with results from optimizing algorithms. We consider the average download performance as evaluation criteria which indicates the aggregate throughput in the community. Our analysis advances the hypotheses that it is possible to increase download performance in BitTorrent communities through better seeder allocation mechanisms.

2 Problem formalization

We represent a BitTorrent community as a triplet (G, L, C) at an instant in time considering the demand in each swarm, in which swarms each user can seed and how many swarms each user can seed. The first two aspects are represented by a directed acyclic bipartite graph $G = (T \cup U, E_l \cup E_s)$, where $T = \{t_1, \ldots, t_m\}$ is the set of swarms currently active in the community, $U = \{u_1, \ldots, u_n\}$ is the set of users in that community, $E_l = \{(u, t) : u \in U, t \in T$ and u is leeching in $t\}$, and $E_s = \{(u, t) : u \in U, t \in T$ and u is able to seed $t\}$. A user is able to seed in a swarm if the user has downloaded the corresponding file in the past and has neither deleted it nor configured the BitTorrent software to stop serving it. We denote $L_i = \{t \mid (u_i, t) \in E_s\}$ as the *library* of user u_i; thus $L := \{L_1, \ldots, L_n\}$.

Besides G and L, we also need to represent how much users can seed. Each user u_i has a seeding capacity (hereafter *capacity* for short) c_i, providing $C := \{c_1, \ldots, c_n\}$. Note that $0 < c_i \leq |L_i|$ holds.

In this paper we investigate a resource allocation problem in which the aim is to find how well a seeding allocation maximizes the mean leeching session throughput. We use the proportion of leechers in swarms to approximate throughput. This metric evaluates an allocation focusing on leeching sessions in a way that the swarms

are characterized by the fixed number of downloaders and by the variable number of seeders in them. Thus, we do not consider which user a leeching session belongs to. In order to formalize this, we introduce further notations. Let a_{ij} be a binary parameter which shows if seeder u_i has the corresponding file of t_j in the library, i.e.,

$$a_{ij} = \begin{cases} 1, & \text{if } t_j \in L_i, \\ 0, & \text{otherwise.} \end{cases}$$

We define the decision variable x_{ij} to denote whether seeder u_i is seeding in swarm t_j:

$$x_{ij} = \begin{cases} 1, & \text{if seeder } i \text{ is seeding in swarm } t_j, \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, $\lambda(t_j)$ denotes the *number of leechers* in swarm t_j, i.e., $\lambda(t_j) = |\{(u, t_j) \in E_l\}|$. Our resource allocation problem is to find

$$\begin{aligned} &\arg\max_x \sum_{j=1}^{m} \frac{\lambda(t_j) \sum_{i=1}^{n} a_{ij} x_{ij}}{\lambda(t_j) + \sum_{i=1}^{n} a_{ij} x_{ij}}, \\ \text{subject to}: \quad &\sum_{j=1}^{m} a_{ij} x_{ij} = c_i \quad (i = 1, \ldots, n), \\ &x_{ij} \in \{0, 1\} \quad (i = 1, \ldots n, j = 1, \ldots m), \end{aligned} \tag{1}$$

which is an integer linear fractional programming problem. The term $\sum_{i=1}^{n} a_{ij} x_{ij}$ gives the *number of actual seeders* in swarm t_j. In the objective function, for each swarm we take the number of actual seeders divided by the size of the swarm (so that this ratio is normalized into (0, 1]) and this ratio is multiplied with the number of leechers in order to weight each swarms when we calculate the global metric (summation of metrics in each swarm). The constraints ensure that the seeders are not seeding more than their capacities.

Our aim is to determine what is the optimal solution of Problem (1) under the typical conditions of real BitTorrent communities. The optimal solution provides us with insights on how good the allocation established by the users of BitTorrent communities following no central instructions is, and how far it is from a random allocation. In order to do so, first a deterministic algorithm is introduced, which is inspired by those used for solving maximum flow problems. Then, we present the datasets obtained from measurements of activities in two BitTorrent communities. The last section gives the numerical results we acquired, followed by concluding remarks.

3 A deterministic algorithm

We develop a deterministic greedy algorithm to maximize seeding allocations according to Problem (1). It takes inspiration from the algorithms for computing the maximum

Algorithm 1 Optimal seeder allocation algorithm

Require: BitTorrent community $((U \cup T, E_s \cup E_l), L, C)$
Ensure: $E' \subseteq E_s$
1: $U^+ \leftarrow \{s_0\} \cup U$; $E_s^+ \leftarrow \{(s_0, u) \mid u \in U \text{ such that } \exists t \in T : (u, t) \in E_s\} \cup E_s$
2: $\forall (s_0, u_i) \in E_s^+ : \xi(s_0, u_i) \leftarrow c_i$
3: $\forall u_i \in U, \forall t_j \in L_i : \xi(u_i, t_j) \leftarrow 1$
4: $\forall (a, b) \in E_s^+ : f(a, b) \leftarrow 0$
5: $E' \leftarrow \emptyset$
6: $Q \leftarrow T$
7: **repeat**
8: $t_{\max} \leftarrow \arg\max_{t \in Q} \left(\frac{\lambda(t)(1+\sum_{(u,t)\in E_s^+} f(u,t))}{1+\lambda(t)+\sum_{(u,t)\in E_s^+} f(u,t)} - \frac{\lambda(t)(\sum_{(u,t)\in E_s^+} f(u,t))}{\lambda(t)+\sum_{(u,t)\in E_s^+} f(u,t)} \right)$
9: $w \leftarrow s_0 t_{\max}$ path such that $\forall (a, b) \in w : \xi(a, b) - f(a, b) > 0$
10: **if** $length(w) = 0$ **then**
11: $Q \leftarrow Q \setminus \{t_{\max}\}$
12: **else**
13: **for all** $(a, b) \in w$ **do**
14: $f(a, b) \leftarrow f(a, b) + 1$
15: $f(b, a) \leftarrow f(b, a) - 1$
16: **end for**
17: **end if**
18: **until** $Q = \emptyset$
19: $E' \leftarrow \{(u, t) \in E_s^+ \mid f(u, t) = 1\}$

flow in a graph [4,5]. In this section a description of the algorithm is given followed by a proof that it finds the optimal allocation for Problem (1).

3.1 Description of the algorithm

Our algorithm takes a BitTorrent community at an instant time as input, transfers it into a flow network graph, on which iteratively selects swarms and increases the number of seeders in them until it reaches the maximum individual values of the sum of the objective function of Problem (1) in all the swarms. That provides the optimal solution of Problem (1), where the feasibility is ensured by the capacity constraints in the flow network.

In the following, we refer to the lines of Algorithm 1 for the formal description. The input of our algorithm is the triplet (G, L, C), which is transfered into a flow network graph $G^+ = (T \cup U^+, E^+, \xi)$. To this end we keep all the existing edges in the graph G, but extending the edges with introducing the *source vertex* s_0 which is then connected to all the seeders in the community (line 1). We define the capacity of the edges in the following way: for the edges between the source vertex s_0 and the seeders the capacity equals to the seeding capacity, while for the other vertices the capacity equals to 1 (line 2–3). This definition of capacities enforces the constraint in the allocation defined in Problem (1). The initial flow f through graph G^+ is set to 0 as it is given in line 4. Note that for the flow function $f : E^+ \to \mathbb{R}$, the property of conservation holds: $\forall u \in U : f(s_0, u) = \sum_{(u,t_i)\in E} f(u, t_i)$.

The algorithm starts processing every swarm in a set Q, initialized with the entire set of swarms T (line 6). The main loop (starting at line 7) finds augmenting paths

w between the source vertex s_0 and all swarms in Q, and keeps running while there are swarms to be processed in Q. For each iteration, a swarm $t_{\max}$ is chosen such that the biggest increment in the objective value is obtained if one more seeder is added to that swarm (line 8). A path w from s_0 to $t_{\max}$ is then constructed (line 9). If there is no such path (i.e., the length of w is zero, as checked in line 10), then $t_{\max}$ is removed from the set Q and the loop condition is evaluated; otherwise, the flow f is updated through the augmenting path w, see lines 13–16. The final allocation is represented by $E' \subseteq E_s$, for which $f(u,t) = 1$ for all $(u,t) \in E_s^+$.

3.2 Optimality and complexity of the algorithm

The objective function of Problem (1) is a sum of non-negative numbers (which we call *sharing ratios* in the following). Thus, this sum is maximized if its components are maximized. Moreover, similarly to the flow-conservation in the classical maximum flow algorithm, when the algorithm increases the sharing ratio in the selected swarm $t_{\max}$ it does not decrease it in any other swarms.

We show now that if there is no more path $w(s_0, t_{\max})$ available for a selected swarm $t_{\max}$, then the swarm $t_{\max}$ reaches its maximum sharing ratio value. To give the proof by transposition, suppose that we did not reach the maximum sharing ratio in swarm $t_{\max}$. This means that it is possible to put at least one more seeder u into this swarm. Thus, there is at least one seeder whose capacity is not saturated, i.e., $\exists u_j : f(s_0, u_j) < c_j$. Therefore, a direct link exists from seeder u_j to the swarm $t_{\max}$, which provides a $w(s_0, t_{\max})$ path, namely the one consists of the edges (s_0, u_j) and $(u_j, t_{\max})$. Note that the algorithm does not check a swarm t anymore if there is no $w(s_0, t)$ path to it. Hence, we conclude that when there are no more swarms left in the set Q, then Algorithm 1 reaches the optimum value for Problem (1).

Regarding the complexity of the algorithm we can see that the main iteration loop has to be done $|L|$ times, the Step 8 can be done in $O(E_s \log E_s)$ time, whereas the path finding in Step 9 takes $O((n+m)^2)$ time.

4 Datasets

This section presents the datasets that inform our analysis and the methodology to extract the necessary information from these datasets.

We use data from two BitTorrent communities: Bitsoup and Filelist. These two communities require users to obtain accounts to participate in the community. In this way, they can track user behavior in all swarms over time. Both communities also employ *sharing ratio enforcement* to promote seeding. This mechanism prevents users who have not uploaded a minimum proportion of the data volume they downloaded from joining new swarms.

Both traces were collected by periodically crawling Web pages in these communities that report statistics. These include, for each user in each swarm, the user name, current uptime and amounts uploaded and downloaded during this uptime. For Bitsoup, these pages were crawled hourly for 64 days; for Filelist, crawling happened on average every six minutes for 93 days. The main characteristics of the resulting

Table 1 Characteristics of the datasets with 95 % CI for averages

Trace	Swarms		Users	Sessions
	Total	Avg. active	Total	Avg. active
Filelist	3,236	512.2 ± 10.2	91,745	32,829.4 ± 672.8
Bitsoup	13,741	6,869.6 ± 30.8	84,007	76,370.3 ± 1,135.5

datasets are summarized in Table 1. We notice that the two communities are significantly different in every aspect. There are more users in Filelist, but much less swarms, and the average number of active swarms, as well as the average number of leeching sessions, are also much lower compared to Bitsoup.

4.1 Estimating user capacities and libraries

We consider the capacity of a user at time τ to be the number of swarms the user is seeding in at τ.

To estimate the contents of the libraries of users, we consider two scenarios that bound the worst- and best-case configurations from the allocation perspective. The worst-case scenario, named *conservative*, considers that a file is in the library of a user at time τ if that user was observed seeding this file both in the past, $[\tau - w_p, \tau]$, and future, $[\tau, \tau + w_f]$. The values w_p and w_f define windows of observation. The second scenario, named *optimistic*, identifies the best-case configuration. In this scenario, a file is in the library of a user at time τ if that user is observed seeding it at least once in the past, $[\tau - w_p, \tau]$. Maintaining these windows constant allows us to perform unbiased comparisons of possible allocations at different times and in different traces.

4.2 Sampling community states

To analyze the seeding allocation in a community, we look at a sample of snapshots of that community taken at random times. All snapshot times τ_i must allow for $\tau_i - w_p$ and $\tau_i + w_f$ to be contained in the sampled trace. The larger the time windows w_p and w_f are, the better the library estimations, but the less space for choosing snapshots and hence the more potential sampling bias.

We address this problem by devising a compromise by setting $w_p = w_f$ to the largest value such that we still have a time period of at least one week for choosing snapshots in both communities. We consider that randomly sampling times in a one week interval accounts for most of the significant fluctuation of BitTorrent users' behavior. As a result, we have $w_p = w_f = 28$ days.

5 Numerical results

This section presents the numerical results obtained on the datasets which were introduced in Sect. 4. From both communities we selected 10 instances and created the

Table 2 Evaluation of the Filelist scenarios

Allocation	Observed	Conservative		Optimistic	
		Random	Optimal	Random	Optimal
f01	3,973.58	3,937.14	4,021.00	2,908.88	4,243.77
f02	3,777.38	3,764.27	3,838.49	3,141.6	4,137.08
f03	3,845.39	3,803.09	3,893.66	2,685.02	4,111.53
f04	4,507.20	4,453.40	4,576.90	3,180.16	4,830.27
f05	4,754.60	4,712.29	4,846.77	3,355.07	5,097.60
f06	4,556.87	4,517.72	4,622.38	3,373.36	4,857.51
f07	2,871.20	2,859.01	2,921.44	2,322.03	3,104.50
f08	4,441.21	4,398.17	4,511.96	2,780.37	4,724.06
f09	4,539.42	4,505.15	4,596.92	2,824.92	4,825.59
f10	4,375.45	4,336.19	4,433.42	2,617.55	4,634.68

Objective function values of Problem (1) are shown

conservative and the optimistic scenarios. The allocations recorded in the community traces serve as the baseline for possible improvements. We call this baseline the *observed* allocation and compare it to *random* allocation, which represents a completely uninformed algorithm, and to the optimal allocation given by our Algorithm 1.

The results obtained for the Filelist community are presented in Table 2. We observe first that the conservative scenario gives space for tiny improvements only, both for random and optimized allocations. This is due to the fact that in this case the variety of possible allocations is very small. On the other hand, considering the optimistic scenario, the possible improvements are much larger. The observed allocations are already 20–45 % better than those of random allocations. The optimal allocations give about 7 % improvements compared to the observed. We conclude that in both scenarios the current allocations are already giving close to optimal allocations, which is mainly due to the fact of large peers-to-swarms ratio.

Albeit the possible improvement is little in terms of the objective value of Problem (1), the actual allocation can be still diverse in different scenarios. To investigate this, Fig. 1 depicts histograms of the swarms' seeder-to-leecher ratio (SLR) for a selected Filelist instance for the observed, conservative and optimistic scenarios. The optimized conservative scenario results in a very similar seeder distribution to that of the observed. On the other hand, the optimized optimistic scenario provides much better distribution: it eliminates most of the very high over-seeding situations as well as decreases the under-seeding ones (the number of swarms with $SLR \leq 1$ got decreased from 34 to 12) and establishes good ratios between 4 and 12.

Regarding the Bitsoup community, the results are more divergent, as we can see in Table 3. Considering the conservative scenario, even random allocation can sometimes give better results than the current ones. Though these are only minor improvements, if any. Comparing the current allocations to the optimal ones, we obtain about 7 % improvements. Turning to the optimistic scenario, which enables larger search space, the current allocation gives about 20 % improvement to the random selection.

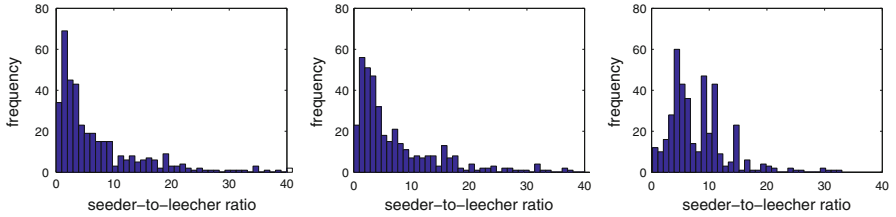

Fig. 1 Histograms of seeder-to-leecher ratios in a FileList instance (f04): observed (*left*), conservative optimized (*middle*), and optimistic optimized (*right*)

Table 3 Evaluation of the Bitsoup scenarios

Allocation	Observed	Conservative		Optimistic	
		Random	Optimal	Random	Optimal
b01	7,244.46	7,328.22	7,758.15	6,112.89	8,579.42
b02	6,881.75	6,939.45	7,350.61	5,806.37	8,251.39
b03	8,770.55	8,742.70	9,224.74	7,346.63	10,197.70
b04	6,943.62	6,975.18	7,360.97	5,896.56	8,258.42
b05	7,800.13	7,717.13	8,099.10	6,592.94	9,020.58
b06	6,730.72	6,764.55	7,137.31	5,625.16	7,985.79
b07	6,935.80	6,887.35	7,240.27	5,719.05	8,103.33
b08	6,121.71	6,191.36	6,557.21	5,126.19	7,375.58
b09	7,046.89	7,091.39	7,460.58	5,916.87	8,379.00
b10	6,819.41	6,930.89	7,315.75	5,813.62	8,203.85

Objective function values of Problem (1) are shown

The optimal allocation provides even better allocations, having further 20 % improvements compared to the current ones. Given that the associated graph of Bitsoup (as it was defined in Sect. 2) is sparser (compared to that of Filelist), giving larger variation of possible allocations, the optimized allocations lead to significantly better average leeching session throughput

Further analysis of the allocations in a particular Bitsoup instance is given in Fig. 2. The histograms of SLR in the observed, conservative optimized and optimistic optimized scenarios are shown. We can observe that, first of all, in the observed allocation, swarms with $SLR \leq 1$ are the most frequent ones, followed by decreasing frequencies of SLR values. For the conservative scenario, the optimized allocation already shows a bit different pattern. Here the swarms with $SLR \leq 1$ have a lower frequency compared to those up to and including 5. From that value on, we obtain a pattern similar to the observed allocation. Finally, for the optimistic optimized scenario, the histogram gives the evidence of a completely different resource allocation structure. SLR values below 3 are much less frequent, and again, we hardly obtain highly overseeded swarms. We conclude that the optimal allocation in both cases (conservative and optimistic) leads not only to better throughput, thus faster average download time, but also more balanced allocation of seeders.

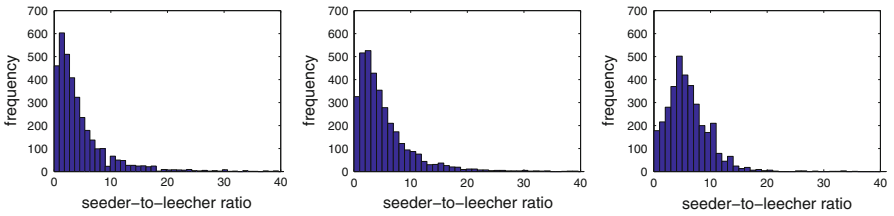

Fig. 2 Histograms of seeder-to-leecher ratios in a BitSoup instance (b01): observed (*left*), conservative optimized (*middle*), and optimistic optimized (*right*)

6 Conclusion

The seeders in BitTorrent file sharing communities can decide in which swarms they want to share their resources. We have shown that optimizing the seeder resource allocation across multiple swarms is equivalent to an integer optimization problem. We evaluated the seeder resource allocation in two communities and compared them to both optimized and random allocations in worst-case and best-case scenarios. Summarizing our findings we conclude that in typical communities, where the number of users is relatively low compared to the number of shared files, it is possible to improve the average throughput as well as decrease the number of under- and over-seeded swarms at the same time.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work was partially supported by the by the Grant TÁMOP-4.2.2/08/1/2008-0008, and by the Future and Emerging Technologies programme FP7-COSI-ICT of the European Commission through project QLectives (grant no.: 231200).

References

1. Andrade, N., Santos-Neto, E., Brasileiro, F., Ripeanu, M.: Resource demand and supply in bittorrent content-sharing communities. Comput Netw **53**(4), 515–527 (2009)
2. Cohen, B.: Incentives build robustness in bittorrent. In: Proceedings of Workshop on Economics of Peer-to-Peer Systems, vol. 6, pp. 68–72 (2003)
3. Dunn, R.J., Gribble, S.D., Levy, H.M.: The importance of history in a media delivery system. In: Proceedings of the Sixth International Workshop on Peer-to-Peer Systems (2007)
4. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)
5. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flowproblem. J. ACM **35**, 921–940 (1988)
6. Meulpolder, M., Pouwelse, J.A., Epema, D.H.J., Sips, H.J.: Bartercast: A practical approach to prevent lazy freeriding in p2p networks. In: Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium (2009)
7. Peterson, R.S., Sirer, E.G.: Antfarm: Efficient content distribution with managed swarms. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, pp. 107–122 (2009)
8. Piatek, M., Isdal, T., Krishnamurthy, A., Anderson, T.: One hop reputations for peer to peer file sharing workloads. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp. 1–14 (2008)

9. Wu, D., Liang, C., Liu, Y., Ross, K.: View-upload decoupling: a redesign of multi-channel p2p video systems. In: Proceedings of IEEE INFOCOM, Rio de Janeiro, Brazil, pp. 1–6 (2009)
10. Yang, Y., Chow, A.L.H., Golubchik, L.: Multi-torrent: a performance study. In: Proceedings of IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, pp. 1–8 (2008)