

ON THE ROLE OF HELPERS IN PEER-TO-PEER FILE DOWNLOAD SYSTEMS: DESIGN, ANALYSIS AND SIMULATION

Jiajun Wang, Chuohao Yeo, Vinod Prabhakaran, Kannan Ramchandran

{junewang, zuohao, vinodmp, kannanr}@eecs.berkeley.edu
Department of EECS, Univ. of California, Berkeley

ABSTRACT

While BitTorrent has been successfully used in peer-to-peer content distribution, its performance is limited by the fact that typical internet users have much lower upload bandwidths than download bandwidths. This asymmetry in bandwidth results in the overall average download speed of a BitTorrent-like file download system to be bottle-necked by the much lower upload capacity. This motivates our approach in which we utilize idle users' spare upload capacity to significantly improve the download speed beyond what can be achieved in a conventional BitTorrent network. We show that this is possible even if these idle users (or helpers) download only a tiny fraction of the file. In fact, in terms of uplink utilization, these helpers can be almost as effective as seeders for medium to large swarms. In this work, we design such a system that is fully compatible with the existing clients who conform to the BitTorrent protocol, analyze its steady-state performance and present simulation results.

1. INTRODUCTION

BitTorrent [1] is a popular Peer-to-Peer (P2P) file sharing protocol that is in wide use today. It has been highly successful as a method of mass content distribution [2] and makes up a large fraction of P2P traffic [3].

The basic idea in BitTorrent is to divide up a large file into equal-sized *pieces* (typically 256KB) and have *peers* (nodes interested in downloading the file) download and upload these pieces to each other simultaneously. Peers contact *trackers*, which are centralized servers, to find other peers to download a file from. *Seeders* are peers who have the entire file, but stay around in the system to upload the file to other peers. BitTorrent uses a particular combination of local-rarest-first piece selection as well as a tit-for-tat rate control

policy to saturate the system upload capacity.

For a BitTorrent-like file sharing system, the system throughput is capped by the minimum of the total system upload bandwidth and the total system download bandwidth [4]. A large population of home Internet users today have asymmetric Internet service, such as ADSL and cable, which makes the total available upload capacity of all downloading peers the bottleneck of a BitTorrent-like file sharing systems.

However, there are other users with spare upload capacity who are not interested in any particular file. We will term such nodes *helpers*. Helpers represent a rich resource of untapped upload bandwidth which can be exploited for increasing the total system upload bandwidth and hence easing the bottleneck. The idea of utilizing helpers free upload capacity was first introduced by Wong [5]. "Tribler" [6] is a social-based peer-to-peer system which provides a mechanism for providing incentive for these helpers to help other users. In this work, we do not address the question of incentive for helpers. Our focus is on developing a systematic approach to optimally utilize such helpers. However, incentive mechanisms like the one in [6] could be adopted for the proposed system.

We develop an efficient mechanism for utilizing the spare upload capacity of helpers while imposing very little download or storage burden on them. In our system, each helper only downloads a small number of random pieces of the file and actively searches for peers who do not possess these pieces to upload to. This way they attempt to fully utilize their upload capacity. In fact, we show that, in terms of upload bandwidth utilization, these helpers are almost as effective as seeders. Our proposed design is a BitTorrent-like system that is fully backward compatible with the original BitTorrent protocol; existing BitTorrent users can

fully realize the gains from these helpers without any modifications on their part. We provide an analysis of its steady-state performance using a simple fluid model of Qiu and Srikant [4], and also present some corrections to the fluid model analysis presented there. We also show simulation results that demonstrate a close match between the analysis and the simulation.

2. RELATED WORK

Guo et. al. [7] briefly explored the idea of inter-torrent collaboration, and described a scheme where peers may download pieces of a file in which they are not interested in exchange for pieces of a file they want to download. In Wong [5], the helpers keep track of the number of times each downloaded piece has been re-uploaded, and uses this information in a download rate-limiting mechanism. However, this is wasteful as we show through both analysis and simulation since a helper who stays in the system for a long time could potentially download a very large portion of the file. In the “Tribler” [6] system, a peer trying to download a file actively recruits helpers to help exclusively with its download. The helpers are directed by the peer in choosing which pieces of the file to download.

Our approach, on the other hand, imposes a much reduced burden on the helpers in terms of the amount of download and the storage space required. Since the helpers download from peers within the system, by reducing the amount of download by the helpers, we improve the over-all system performance.

3. SYSTEM DESCRIPTION

In this section, we describe a system compatible with existing clients conforming to a BitTorrent protocol.

As stated before, helpers are not interested in the file being downloaded. Thus it is unnecessary and undesirable for them to download the entire file or a significant portion of the file. A helper would not want to waste much of its storage space to store a file that it is not interested in. More importantly, by minimizing the amount the helpers have to download, we can leave most of the available download bandwidth to the peers who are actually interested in the file.

We propose that helpers stop downloading after obtaining a small number of pieces, k , of the file. If the total number of pieces in the file is N , we show below that even with $k \ll N$, the helpers’ efficiency in distributing the file is almost as good as if they have the

entire file when the swarm size is not too small. The intuition is that when a helper has only a small number of pieces, it is of limited use to any one peer, but in a swarm, it is useful to a large number of the peers. As long as the helper is successful in finding these peers, it can fully utilize its upoad capacity. Our simulations bear this out.

3.1. Helper protocol

A helper joins a swarm just like a regular peer. The download mechanism of a helper is the same as that of a regular peer except

- a helper only downloads k pieces of the file, with k being a fixed small number;
 - helpers are not allowed to download from each other.
- We will explain in detail the reason behind these design choices in Section 3.2.

In choosing which k pieces to download, different strategies can be adopted. We found that both random and local-rarest-first[1] mechanisms to be equally effective. Once a helper finishes downloading k pieces of the file, it stops downloading. We call such a node a *microseed*. At this time, it reports to the tracker and obtains a list of regular peers who have downloaded less than a certain fraction of the file¹. This is motivated by the fact that a helper is useful to a peer if the helper has at least one piece of the file that the peer does not have; and a helper is more likely to be useful to a peer with a small portion of the file. Helpers (including microseeds) implement the same choking/unchokeing procedure that is adopted by any BitTorrent client with one exception: once a microseed sees that a neighbor has already obtained all the pieces it has to offer, it will automatically disconnect from the neighbor. We note that in this protocol, we require that helpers be aware of each other when connected and that trackers be aware of helpers. As mentioned earlier, there is no need for existing peers to be modified.

3.2. Efficiency of helpers

We first show that the average probability that a microseed has at least one piece that a peer does not have is $\frac{k}{k+1}$ if $k \ll N$, assuming the number of pieces a random peer has is uniformly distributed in $\{0, 1, \dots, N-1\}$. We know that

$$\begin{aligned} & \mathbb{P}(\text{Peer } i \text{ needs at least 1 piece from Microseed } j) \\ &= 1 - \mathbb{P}(\text{Peer } i \text{ needs no piece from Microseed } j) \end{aligned}$$

¹If this list is too short, the tracker will pad it with peers who have downloaded more.

Let n_i denote the number of pieces a random peer i has acquired. Since n_i is uniformly distributed in $\{0, 1, \dots, N - 1\}$, we have

$$\begin{aligned}
& \mathbb{P}(\text{Peer } i \text{ needs no piece from Microseed } j) \\
&= \mathbb{P}(\text{Peer } i \text{ has all pieces of Microseed } j) \\
&= \sum_{n_i=k}^{N-1} \frac{1}{N} \mathbb{P}(\text{Peer } i \text{ has all pieces of Helper } j | n_i) \\
&= \sum_{n_i=k}^{N-1} \frac{1}{N} \cdot \frac{\binom{N-k}{n_i-k}}{\binom{N}{n_i}} \rightarrow \frac{1}{k+1} \text{ if } N \gg k \quad (1)
\end{aligned}$$

If a helper has m neighboring peers, the probability that at least one of its neighbors needs at least a piece from the helper is $1 - (\frac{1}{k+1})^m$. Since this value is close to 1, the upload capacity of the helper is almost always utilized even if it only downloads a very small number of pieces.

In practice, however, in order to reduce the amount of connection and query overhead, we need to make sure that every peer that a helper connects to will need at least one piece from the helper with high probability (say p). For any helper, the probability of finding such a peer is high if it looks among peers who have finished only a small portion of the download. If the tracker keeps track of the rough download progress of all the peers in the swarm², then when a helper queries the tracker for a list of peers to help, the tracker should reply with a list of peers who have only finished downloading a limited number of pieces (say l). There is not much incentive for a peer to take advantage of the system by mis-reporting its download progress since the probability of being helped by a helper is low once it has downloaded a large fraction of the file. Given a target probability p and the number of pieces k a microseed has, we can solve for l from $1 - \frac{\binom{N-k}{l-k}}{\binom{N}{l}} \geq p$.

As mentioned earlier, we disallow helpers to download from each other. This is because we would like the helpers (microseeds) to have a collection of uniformly distributed pieces. Intuitively, if all the helpers have the same pieces, their helpfulness would be much reduced. Our simulations show that when helpers are allowed to download from one another, they are likely to have pieces that are not as evenly distributed. This is especially true when helpers enter a swarm at a relatively high rate.

²Note that the BitTorrent protocol does allow the tracker to keep track of the download progress of the peers [1].

4. SYSTEM ANALYSIS USING FLUID MODEL

We model the steady-state behavior of the system with homogeneous peers using the simple “fluid model” proposed in [4] and make some important corrections.

Following the model in [4], we have the following assumptions:

- The peer arrival process is Poisson with rate λ .
- The peers are homogeneous and have the same upload bandwidths μ and the same download bandwidths c . Here we assume $\mu < c$.
- The peers download at the same rate when they are downloading. This rate is the total upload bandwidth from all the participating peers divided up equally among the downloading peers. Thus we assume that the instantaneous download rate is the same for all the downloading users.

We present the analysis for the case where the upload-bandwidth is the constraint as it is the scenario of interest. We define the following quantities (as in [4]):

- μ Upload bandwidth of the peers (b/s)³
- λ Arrival rate of peers (users/s)
- $x(t)$ Number of peers downloading at time t
- $y(t)$ Number of seeders in the system at time t
- θ Patience factor of peers (1/s). If the peer does not finish downloading after at most a random interval with an exponential distribution (independent across peers) with mean $1/\theta$, it aborts the download.
- γ Patience factor of seeders (1/s). Having finished the download, each peer remains in the system for a further random interval with an exponential distribution with mean $1/\gamma$, independent across peers.
- η Efficiency factor of peers who are downloading. This is the average probability that a downloading peer has at least one piece that one of the peers it is connected to does not have.

In addition, we define $w(t)$ (b/s) as the bandwidth “wasted” by peers who abort before downloading the whole file. This wastage was not accounted for in the model in [4]. By leaving out this term, the bandwidth consumed by the aborting peers was also counted towards contributing to seeder creation leading to inflated performance. The discrepancy can be significant in cases where a significant number of peers abort without downloading the whole file.

³All bandwidths are normalized so that the size of the file can be taken to be unity.

For the system without any helpers, the total upload rate available is given by $\mu(\eta x(t) + y(t))$. Since the file size is normalized to unity, $\mu(\eta x(t) + y(t)) - w(t)$ is the rate at which peers are finishing their downloads and turning into seeders. Hence we have

$$\frac{dx(t)}{dt} = \lambda - \theta x(t) - [(\mu(\eta x(t) + y(t))) - w(t)] \quad (2)$$

$$\frac{dy(t)}{dt} = [(\mu(\eta x(t) + y(t))) - w(t)] - \gamma y(t) \quad (3)$$

If we use x and y to denote the steady-state values of $x(t)$ and $y(t)$ respectively, the time taken by a peer to download the entire file in steady-state is given by

$$T_{\text{dl}} = \frac{1}{\text{steady-state peer-download-rate}} = \frac{1}{(\mu(\eta x + y))/x}. \quad (4)$$

The wasted bandwidth under steady-state can then be computed as

$$w = \theta \frac{\int_0^{T_{\text{dl}}} p_{\theta}(t) t \mu(\eta x + y) dt}{\int_0^{T_{\text{dl}}} p_{\theta}(t) dt}, \quad (5)$$

where $p_{\theta}(t) = \frac{1}{\theta} \exp(-\theta t)$, $t \geq 0$ is the exponential pdf with mean $1/\theta$.

To obtain the steady-state solution, we set the steady-state conditions $\frac{dx(t)}{dt} = \frac{dy(t)}{dt} = 0$ and solve (2) to (5) for x and y . From this we obtain the steady-state average download time T_{dl} using (4). Note that these equations can be solved only numerically.

In cases where the wastage term w is small enough to be ignored, we can analytically solve the equations above. The resulting T_{dl} is given by

$$T_{\text{dl}} = \frac{\gamma/\mu - 1}{\gamma\eta}. \quad (6)$$

As pointed out earlier, [4] ignores the effect of w . Furthermore, the expression of T_{dl} in [4] was in error⁴, and hence differs from Eqn. (6). However, the conclusion in [4] that T_{dl} is independent of the peer arrival rate λ remains true (whenever the effect of wastage can be ignored).

To analyze the system with helpers, we introduce

- μ_h Upload bandwidth of the helpers (b/s)
- λ_h Arrival rate of peers (users/s)

⁴This is because the expression given in [4] is in fact the time spent downloading by a peer averaged over the peers who finish the download *as well as* the peers who abort without finishing their download, and not the time taken to download the file averaged over only the peers who successfully download the entire file. This results in an overly optimistic value for the download time. The error is the result of an inappropriate use of Little's law.

- $x_h(t)$ Number of helpers downloading at time t
 - $y_h(t)$ Number of microseeds in the system at time t
 - $w_h(t)$ Bandwidth “wasted” by helpers who abort before downloading k pieces (b/s)
 - ε The proportion of the total upload bandwidth that is consumed by the downloading peers. In our system, this is just $x_h(t)/(x(t) + x_h(t))$.
 - θ_h Patience factor of helpers (1/s). Each helper remains in the system for a random interval with an exponential distribution (independent across helpers) with mean $1/\theta_h$ irrespective of whether they finish downloading their portion or not.
 - η_h Efficiency factor of helpers who have finished downloading their portion (defined similar to η)
 - ρ The fraction of the file a helper download (k/N).
- Then the fluid model can be written as

$$\frac{dx(t)}{dt} = \lambda - \theta x(t) - [\varepsilon(\mu(\eta x(t) + y(t)) + \mu\eta_h y_h(t)) - w(t)]$$

$$\frac{dy(t)}{dt} = [\varepsilon(\mu(\eta x(t) + y(t)) + \mu\eta_h y_h(t)) - w(t)] - \gamma y(t)$$

$$\frac{dx_h(t)}{dt} = \lambda_h - \theta_h x(t) - \frac{(1 - \varepsilon)(\mu(\eta x(t) + y(t)) + \mu\eta_h y_h(t) - w_h(t))}{\rho}$$

$$\frac{dy_h(t)}{dt} = \frac{(1 - \varepsilon)(\mu(\eta x(t) + y(t)) + \mu\eta_h y_h(t) - w_h(t))}{\rho} - \gamma_h y(t)$$

Again, to study the steady-state behavior, we set these to zero and solve for the average number of peers, seeders, helpers, and microseeds. The download time is now given by $T_{\text{dl}} = 1/[\varepsilon(\mu(\eta x(t) + y(t))) + \mu\eta_h y_h(t)]$.

In the next section we show numerical solutions obtained from this analysis for our different simulation setups along with our simulation results.

5. SIMULATION RESULTS

We have implemented a simulator for both the BitTorrent system and our proposed helper system for BitTorrent. Among other things, the simulator explicitly implements what we feel are the defining features of the protocol, namely: (i) local-rarest-first piece selection; and (ii) tit-for-tat policy in conjunction with choking/unchoking (including optimistic unchoking).

We first study the accuracy of the fluid model by comparing average download time computed through the fluid model closely to that obtained from simulations. We use two different setups. Setup 1 mimics a relatively small file with a small-sized swarm while Setup 2 has a large file with a medium-sized swarm. The parameters for the 2 setups are shown in Table 1.

Table 2 shows that under both setups, the average download time, the number of peers (excluding seeds),

	Setup 1	Setup 2
$\lambda(\text{peer/s})$	0.2	0.2
$\theta(1/\text{s})$	0.000625	0.000625
$\mu(\text{kbps})$	512	512
$c(\text{kbps})$	2048	2048
$\gamma(1/\text{s})$	0.0025	0.00025
$\lambda_h(\text{peer/s})$	0.05	0.1
$\theta_h(1/\text{s})$	0.00125	0.00025
$\mu_h(\text{kbps})$	512	512
$c_h(\text{kbps})$	2048	2048
file size (MB)	100 ($N = 400$ pieces)	1000 ($N = 1000$ pieces)
k	8	10

Table 1. Parameters for 2 different setups.

and the number of seeds in the system in steady state is accurately estimated by the fluid model.

Setup 1	Helper	DL time	# of peers	# of seeds
Model	No	1344.83	181.924	34.519
Model	Yes	1090.24	158.11	40.47
Simulation	No	1389.29	187.15	35.23
Simulation	Yes	1161.02	163.30	39.29
Setup 2	Helper	DL time	# of peers	# of seeds
Model	No	13448.3	1819.24	345.19
Model	Yes	10902.4	1581.1	404.726
Simulation	No	13551.7	1829.52	350.03
Simulation	Yes	11309.7	1622.74	393.30

Table 2. The average download time, average number of peers (not including seeds) and average number of seeds obtained through the fluid model matches closely with simulation result with and without helpers.

Figure 1 shows that under Setup 1, the system throughput is extremely close to the total available upload bandwidth, indicating that the helpers' upload bandwidths are being utilized fully even though they only download $k = 8$ out of 400 pieces of the file. This shows that helpers are almost as effective as seeds even if helpers download only a small number of pieces. In fact, we show in Figure 2 that if helpers download too many pieces, it actually hurts the system performance because helpers are not interested in the content themselves and the amount of system download bandwidth they use to acquire the required number of pieces is actually wasted. However, we cannot set k arbitrarily low either, as explained in Section 3.2.

Figure 3 shows the CDF of peer download time of 4 different systems under Setup 1:

System 1: An upper-bound setup where helpers join the swarm with k random pieces preloaded.

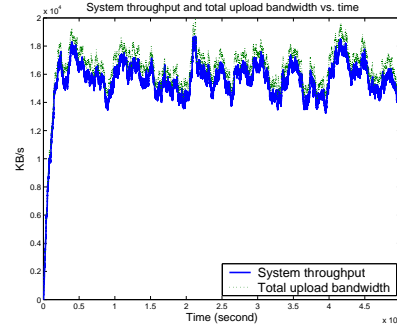


Fig. 1. Throughput and total upload bandwidth of the system over time. It can be observed that throughput is kept very close to total upload bandwidth indicating that the helpers' upload bandwidths are being fully utilized.

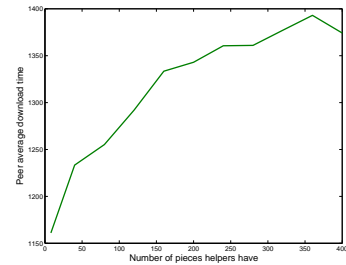


Fig. 2. Avg. download time vs. # of pieces (k) helpers download.

System 2: System with regular helpers.

System 3: System with a small number (x_s) of extra seeders such that $x_s \cdot N = x_h \cdot k$ where x_h is the number of helpers (including microseeds), N is the number of pieces in the file.

System 4: An unaltered BitTorrent system.

The comparison between System 2 and 4 shows that a system with helpers have a much lower average peer download time. The comparison between System 2 and 3 shows that it is much more efficient to have a large number of helpers each with a small number of pieces than to have a small number of extra seeds. The comparison between System 1 and 2 shows that by limiting the number of pieces helpers download, very little system resource is wasted.

We would like to point out that if the helpers spend too little time in the system and spend most of it downloading the required number of pieces, the overall system performance could actually be hurt⁵. However, if the tracker can estimate the helper departure rate,

⁵Note that it is the total amount of time they spend in the system that matters. The system performance will not be adversely affected even if the helpers log on and off frequently, but preserve the small number of pieces they downloaded across sessions.

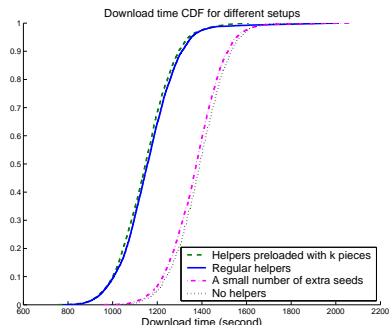


Fig. 3. CDF of peer download time in 4 different systems. System 1: An upper-bounding setup where helpers join the swarm with k random pieces preloaded. System 2: System with regular helpers. System 3: System with a small number (x_s) of extra seeds such that $x_s \cdot N = x_h \cdot k$ where x_h is the number of helpers (including microseeds), N is the number of pieces in the file. System 4: An unaltered BitTorrent system

it could use the fluid model to numerically solve for an upper bound to the number of pieces a helper can download in order not to hurt the system by setting the average peer download time with helpers to that when there are no helpers. Also, the tracker may turn away helpers when the helper arrival rate exceeds the rate required to fill up regular peers' download pipes.

Finally, we show results with a heterogenous crowd. We simulated three classes of users with different upload/download bandwidths. 25% of the peers have 512:128 Kbps, 50% of the peers have 2048:512 Kbps and the other 25% have 8:1 Mbps. 75% helpers have 2048:512 Kbps and the rest have 8:1 Mbps. Figure 4 shows that helpers improve the download time for all the users, but are most helpful to peers with slower upload/download speed. This is expected as these peers are not favored by the optimistic unchoking process and have a much slower start up phase without helpers.

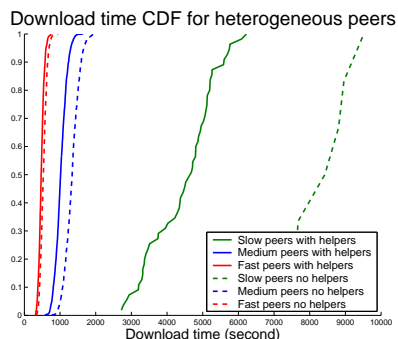


Fig. 4. CDF of peer download time for users with different upload/download speed.

6. CONCLUSIONS

We proposed a system for using the spare upload capacities of idle users (helpers), for improving the download performance of peers in a BitTorrent-like system. Our scheme is fully backwards-compatible with the existing BitTorrent clients, and only requires slight modifications to the existing BitTorrent infrastructure. In our system, helpers only download a tiny fraction of the file of interest. This presents a low storage cost to the helpers, and is minimally taxing on the swarm. We showed that with this design, the helpers' upload bandwidth can be fully utilized. We also presented, based on a fluid model, an analysis of our system which can be used for determining the system parameters and predicting the system behavior. Our simulation results matched the predictions of the analysis and demonstrated the benefits of the proposed system. We are currently implementing our system on a real network environment to further validate its performance.

Acknowledgement: we would like to thank Charlie Lin of UC Berkeley for his help with the simulations.

7. REFERENCES

- [1] B. Cohen, "Incentives Build Robustness in BitTorrent," *Workshop on Economics of Peer-to-Peer Systems*, vol. 6, 2003.
- [2] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five Months in a Torrents Lifetime," *Passive and Active Measurements*, vol. 2004, 2004.
- [3] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The Bittorrent P2P file-sharing system: Measurements and analysis," in *4th Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, Feb 2005.
- [4] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *SIGCOMM '04*.
- [5] J. Wong, "Enhancing collaborative content delivery with helpers," *Master's thesis, Univeristy of British Columbia*, Nov 2004.
- [6] J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. Epema, M. Reinders, M. van Steen, and H. Sips, "Tribler: A social-based Peer-to-Peer system," *The 5th International Workshop on Peer-to-Peer Systems*, 2006.
- [7] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, analysis and modeling of bittorrent-like systems," in *Proc. ACM IMC 2005*, October 2005.