To appear in Performance Evaluation Review, Eighth Workshop on Mathematical Performance Modeling and Analysis (MAMA 2006) Issue

Analysis of Peer-to-Peer File Dissemination

Jochen Mundinger[*], Richard Weber[†] and Gideon Weiss[‡]

1 Introduction

In recent years, overlay networks have proven a popular way of disseminating potentially large files from a single server S to a potentially large group of N end users via the Internet. A number of algorithms and protocols have been suggested, implemented and studied. In particular, much attention has been given to peer-to-peer (P2P) systems such as BitTorrent [5], Slurpie [18], SplitStream [4], Bullet [10] and Avalanche [6]. The key idea is that the file is divided into M parts of equal size and that a given user may download any one of these – or, for Avalanche, linear combinations of these – either from the server or from a peer who has previously downloaded it.

However, performance analysis of P2P systems for file dissemination has typically been limited to comparing one system relative to another and typically been realized by means of simulations and measurements. We give the minimal time to fully disseminate the file of M parts from a server to N end users in a centralized scenario. In the scheduling literature this completion time is referred to as *makespan*. We thereby provide a lower bound which can be used as a performance benchmark for any P2P file dissemination system. We also investigate the part of the loss in efficiency that is due to the lack of centralized control in practice. Using simulation as well as direct computation, we show that even a simple and natural randomized strategy disseminates the file in an expected time that grows with N in a similar manner to the minimal time achieved with a centralized controller. This suggests that the performance of necessarily decentralized P2P file dissemination systems should still be close to our performance bound.

In this paper, we extend our results from [14] by providing a closed form solution for the makespan in the case of equal upload capacities. We complement them by investigating decentralized solutions. In [19] the authors also consider problems concerned with the service capacity of P2P networks. They give a heuristic argument for the makespan with equal upload capacities when N is of the simple form $2^n - 1$. In [16] a fluid model for BitTorrent-like networks is introduced and studied, also looking at the effect of incentive mechanisms to address free-riding. Link utilization and fairness are issues in [3]. In [12], also motivated by the BitTorrent protocol and file swarming systems in general, the authors consider a probabilistic model of coupon replication systems. Multi-torrent systems are discussed in [7]. There is other related work in [17].

2 The Uplink-Sharing Model

Our analysis is based on the uplink-sharing model [13, 14], an abstract model focusing on the important features of P2P file dissemination. Underlying the file dissemination system is the Internet. Thus, each user can connect to every other user and the network topology is a complete graph. The server S has upload capacity C_S and the N peers have upload capacities $C_1, \ldots, C_N$, measured in megabytes per second (MBps). Once a user has received a file part it can participate subsequently in uploading it to its peers. It is supposed that, in principle, any number of users can simultaneously connect to the server or another peer, the available upload capacity being shared equally amongst the open connections. Taking the file size to be 1 MB, this means that if n users try simultaneously to download a part of the file (of size $1/M$) from the server then it takes n/MC_S seconds for these downloads to complete. Observe that the rate at which an upload takes place can both increase and decrease during the time of that upload (varying according to the number of other uploads with which it shares the upload capacity), but we assume that uploads are not interrupted until complete. In fact, Lemma 2.1 below shows that the makespan is not increased if we restrict the server and all peers to carry out only a single upload at time. Users are permitted to download more than one file part simultaneously, but these must be from different sources; only one file part may be transferred from one user to another at the same time. More complicated interactions are ignored and it is assumed that the upload capacities, $C_S, C_1, \ldots, C_N$, impose the only constraints on the rates at which file parts can be transferred between peers which is a reasonable assumption if the underlying network is not overloaded. Finally, it is assumed that rates of uploads and downloads do not constrain one another.

Note that the download rates are assumed to be unconstrained and this might be considered unrealistic. However, Theorem 3.3 below shows that if the upload capacities are equal then additional download capacity constraints do not increase the minimum possible makespan, as long as these download capacities are at least as

[*]EPFL-IC-LCA, BC203, Station 14, CH-1015 Lausanne, Switzerland
[†]Statistical Laboratory, Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WB, UK
[‡]Department of Statistics, University of Haifa, Mount Carmel 31905,Israel

big. This is usually the case in practice. Typically, N is of the order of several thousands and the file size is up to a few gigabytes (GB), so that there are several thousand file parts of size 1/4 MB each.

Finding the minimal makespan looks potentially very hard as upload times are interdependent and might start at arbitrary points in time. However, the following two results help simplify it dramatically.

Lemma 2.1 *In the uplink-sharing model the minimal makespan is not increased by restricting attention to schedules in which the server and each of the peers only carry out a single upload at a time.*

Lemma 2.2 *In the uplink-sharing model the minimal makespan is not increased by restricting attention to schedules in which uploads start only at times that other uploads finish or at time* 0*.*

3 Solution for General Capacities

In this section, we summarize our previous results from [14].

Theorem 3.1 *The minimal makespan problem amongst users of general upload capacities can be solved via a mixed integer linear program MILP formulation.*

MILPs are well-understood and there exist efficient computational methods [2] that have been implemented in many commercial optimization libraries such as OSL or CPLEX. As the numbers of variables and constraints in the MILP grows exponentially in N and M, this approach is not practical for large N and M, but the MILP can also be used to obtain a bounded approximation for the solution.

Further insight is provided by considering special choices for N, M and C_i. In particular, for a large number of file parts M, the problem is approximated by a fluid limit problem, in which the file is infinitely divisible. A surprisingly simple result is obtained that can be generalized even further. Instead of there being just one designated server with a file to disseminate, suppose each user i has a file of size $F_i \geq 0$ that is to be disseminated to all other users. Seeing that there is no longer a particular server and now everything is symmetric, we change notation for the rest of this section so that there are $N \geq 2$ users $1, 2, \ldots, N$ in all. Moreover, let $F = \sum_{i=1}^{N} F_i$ and $C = \sum_{i=1}^{N} C_i$.

Theorem 3.2 *In the fluid limit, the minimal makespan is*

$$T^* = \max\left\{\frac{F_1}{C_1}, \frac{F_2}{C_2}, \ldots, \frac{F_N}{C_N}, \frac{(N-1)F}{C}\right\} \tag{3.1}$$

and this can be achieved with a two-hop strategy, i.e., one in which the file at the server is uploaded to user i, either directly from the server, or via at most on intermediate user j.

Even though files are not infinitely divisible, often in practice $M >> \log(N)$ and this turns out to be sufficient for the fluid solution to be a good approximation. Thus, the MILP solution is suitable for small values of M and the fluid solution is suitable for typical and large values of M.

4 Solution for Equal Capacities

Assuming upload capacities are equal, we can say even more. In this section, we provide a closed form solution for the makespan in this case. We first consider the simultaneous send/receive broadcasting model [8] in which the server and all users have upload capacity of 1. Theorem 3.1 agrees with a result by Bar-Noy, Kipnis and Schieber [1], who obtained this as a by-product of their result on the bidirectional telephone model. However, they required pairwise matchings in order to apply their results for the telephone model. As a result, their algorithm differs for odd and even N, and it is substantially more complicated, to describe and prove to be correct, than the one we present within our proof. It also agrees with a result of Kwon and Chwa [11], but they, too, do not give a direct proof, but use broadcasting in hypercubes.

Theorem 4.1 *In the simultaneous send/receive broadcasting model, with all upload and download capacities equal to 1, the minimum number of rounds is $M + \lfloor \log_2 N \rfloor$ each round taking up $1/M$ units of time. Equivalently, for all M, N, the minimal makespan is*

$$T = 1 + \frac{\lfloor \log_2 N \rfloor}{M}. \tag{4.2}$$

The solution of the simultaneous send/receive broadcasting model problem then gives the solution of our uplink-sharing model when all capacities are the same. Moreover, in the proof of Theorem 4.1 we explicitly give an optimal schedule which also satisfies the constraints that no peer downloads more than a single file part at a time.

Theorem 4.2 *Consider the uplink-sharing model with all upload capacities equal to* 1*. The minimal makespan is given by* (4.2)*, the same as in the simultaneous send/receive model with all upload capacities equal to* 1*.*

Theorem 4.3 *In the uplink-sharing model with all upload capacities equal to 1, constraining the peers' download rates to 1 does not further increase the minimal makespan.*

5 Decentralized Solution

In order to give a lower bound on the minimal makespan and to obtain a performance benchmark, we have been assuming a centralized controller. We now consider a very simple and natural randomized strategy and investigate the loss in performance that is due to the lack of centralized control. If such a simple strategy does not lose much compared to the minimal makespan, then more sophisticated strategies will lose even less and the performance of necessarily decentralized P2P file dissemination systems can be expected to be close to our performance bound.

Let us start with the special case $M = 1$. It makes sense to assume that each peer knows the number of file parts M and the address of the server. However, a peer might not know N, the total number of peers, nor its peers' addresses, nor if they have the file, nor whether they are at present occupied uploading to someone else.

We consider two different information scenarios. In the first one, *List*, the number of peers holding the file and their addresses are known. In the second one, *NoList*, the number and addresses of all peers are known, but it is not known which of them currently hold the file. Thus, in *List*, downloading users choose uniformly at random between the server and the peers already having the file. In *NoList*, they choose uniformly amongst all the peers. If a peer receives a query from a single peer, he uploads the file to that peer. If a peer receives queries from multiple peers, he chooses one of them uniformly at random. The others remain unsuccessful in that round.

For the problem with one server and N users we have carried out 1000 independent simulation runs[1] for each $N = 2, 4, \ldots, 2^{25}$. We found that the achieved expected makespan appears to grow as $a + b \times \log_2 N$, and so we fitted the linear model $y_{ij} = \alpha + \beta x_i + \epsilon_{ij}$, where y_{ij} is the makespan for $x_i = \log_2 2^i$, obtained in run j, $j = 1, \ldots, 1000$. We obtain the following results. For *List*, the regression analysis gives a good fit, with Multiple R-squared value of 0.9975 and significant p- and t-values. The makespan increases as

$$1.1392 + 1.1021 \times \log_2 N\,. \tag{5.3}$$

For *NoList*, there is more variation in the data than for *List*, but, again, the linear regression gives a good fit, with Multiple R-squared of 0.9864 and significant p- and t-values. The makespan increases as

$$1.7561 + 1.5755 \times \log_2 N\,. \tag{5.4}$$

As expected, the additional information for *List* leads to a significantly lesser makespan, in particular the log-term coefficient is significantly smaller. Note that *List* achieves a makespan that is very close to the centralized optimum of $1 + \lfloor \log_2 N \rfloor$: It is only suboptimal by about 10%. Hence even this simple randomized strategy performs well in both cases and very well when state information is available.

In fact, it is possible to compute the mean makespan analytically by considering a Markov Chain on the state space $0, 1, 2, \ldots, N$, where state i corresponds to i of the N peers having the file. The resulting formula is rather complicated, but can be evaluated exactly using arbitrary precision arithmetic on a computer. Computation times are long, so to keep them shorter we only work out the transition probabilities exactly. Hitting times are then computed in double precision. Even so, computations are only feasible up to $N = 512$ with our equipment, despite repeatedly enhanced efficiency. Moreover, the difference to the simulated values is small without any apparent trend confirming the simulation results. This suggests that simulations are the more efficient approach to our problem and we shall stick to simulations when investigating the general case of M file parts.

In the general case, we repeat the simulations for *List* for various values of M. We carried out 100 independent runs for each $N = 2, 4, \ldots, 2^{15}$ and fitted the same linear model as before.

Table 1 summarizes the simulation results. The Multiple R-squared values indicate a good fit, although the fact that these decrease with M suggests there may be a finer dependence on M or N. In fact, we obtain a better fit using Generalized Additive Models. However, our interest here is not in fitting the best possible model, but to compare the growth rate with N to the one obtained in the centralized case in Section 4. From the diagnostic plots we note that the actual performance for large N is better than given by the regression line, increasingly so for increasing M. In each case, we obtain significant p- and t-values. The regression $0.7856 + 1.1520 \times \log_2 N$ for $M = 1$ does not quite agree with $1.1392 + 1.1021 \times \log_2 N$ found in (5.3). It can be checked, by repeating the analysis there for $N = 2, 4, \ldots, 2^{15}$ that this is due to the different range of N. Thus, our earlier result of 1.1021 might be regarded more reliable, being based on N ranging up to 2^{25}.

We conclude that, as in the centralized scenario, the makespan can be reduced significantly in a decentralized scenario even with a simple random strategy. However, by comparing the second and fourth columns of Table 1, as M increases the achieved makespan compares less well relative to the centralized minimum of $1 + (1/M)\lfloor \log_2 N \rfloor$.

6 Further Work

It would now be very interesting to compare dissemination times of the various overlay networks directly to our performance bound. A mathematical analysis of the protocols is rarely tractable, but simulations or measurements such as in [9] and [15] for the BitTorrent protocol can be carried out in an environment suitable for this comparison.

[1] As many as 1000 runs were required for comparison with the results obtained from direct computation later in this section, mainly because the makespan always takes integer values.

M	Fitted makespan	Mult. R-sqd.	$1/M$
1	$0.7856 + 1.1520 \times \log_2 N$	0.9947	1.000
2	$1.3337 + 0.6342 \times \log_2 N$	0.9847	0.500
3	$1.4492 + 0.4561 \times \log_2 N$	0.9719	0.333
4	$1.4514 + 0.3661 \times \log_2 N$	0.9676	0.250
5	$1.4812 + 0.3045 \times \log_2 N$	0.9690	0.200
8	$1.4907 + 0.2113 \times \log_2 N$	0.9628	0.125
10	$1.4835 + 0.1791 \times \log_2 N$	0.9602	0.100
15	$1.4779 + 0.1326 \times \log_2 N$	0.9530	0.067
20	$1.4889 + 0.1062 \times \log_2 N$	0.9449	0.050
50	$1.4524 + 0.0608 \times \log_2 N$	0.8913	0.020

Table 1: Simulation results in the decentralized *List* scenario for various values of M and log-term coefficients in the centralized optimum (cf. Theorem 4.1).

In practice, splitting the file and passing on extra information has an overhead cost. Although this is considered to be small (less than 0.1% [5]), this could be investigated in more detail. There is a lot of freedom in the fluid solution used in the proof of (3.1) in Section 3. We have focussed on solutions that satisfy certain symmetry properties, but it would be interesting to consider others.

Finally, one might want to consider a dynamic setting with peers arriving and perhaps leaving when they have completed the download of the file, cf. [16], [19]. One might also want to extend our model to consider users who prefer to free-ride and do not wish to contribute uploading effort.

References

[1] A. Bar-Noy, S. Kipnis, and B. Schieber. Optimal multiple message broadcasting in telephone-like communication systems. *Discrete Applied Mathematics*, 100:1–15, 2000.

[2] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, 1997.

[3] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Some observations on bitTorrent performance. *Performance Evaluation Review*, 33(1):398–399, 2005.

[4] C. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, 2003.

[5] B. Cohen. Incentives build robustness in BitTorrent. *Proceedings of the Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA*, 2003.

[6] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *IEEE INFOCOM 2005*, March 2005.

[7] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proceedings of Internet Measurement Conference 2005 (IMC 2005)*, October 2005.

[8] J. Hromkovic, R. Klasing, B. Monien, and R. Peine. *Combinatorial Network Theory (F. Hsu and D.-Z. Du, Eds.)*, chapter Dissemination of Information in Interconnection Networks (Broadcasting and Gossiping), pages 125–212. Kluwer Academic Publishers, 1995.

[9] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garcés-Erice. Dissecting BitTorrent: Five months in a torrent's lifetime. *Passive and Active Measurements 2004*, 2004.

[10] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using and overlay mesh. In *19th ACM Symposium on Operating Systems Principles (SOSP'03)*, 2003.

[11] C.-H. Kwon and K.-Y. Chwa. Multiple message broadcasting in communication networks. *Networks*, 26:253–261, 1995.

[12] L. Massoulié and M. Vojnović. Coupon replication systems. In *ACM Sigmetrics 2005*, June 2005.

[13] J. Mundinger and R. R. Weber. Efficient content distribution using peer-to-peer technology. In *C. Griwodz, T. P. Plagemann and R. Steinmetz. 04201 Abstracts Collection – Content Distribution Infrastructures. Dagstuhl Seminar Proceedings*, 2006.

[14] J. Mundinger, R. R. Weber, and G. Weiss. Analysis of peer-to-peer file dissemination amongst users of different upload capacities. *Poster at Performance 2005*, October 2005.

[15] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The Bittorrent P2P file-sharing system: Measurements and analysis. In *4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, February 2005.

[16] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *Proc. ACM SIGCOMM*, September 2004.

[17] K. K. Ramachandran and B. Sikdar. An analytic framework for modeling peer to peer networks. *IEEE INFOCOM 2005*, 2005.

[18] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. *IEEE INFOCOM 2004*, 2004.

[19] X. Yang and G. de Veciana. Service capacity of peer to peer networks. *IEEE INFOCOM 2004*, 2004.