Analysis of Peer-to-Peer File Dissemination amongst Users of Different Upload Capacities

J. Mundinger[1] , R. R. Weber[1] and G. Weiss[2]

1 Introduction

In recent years, overlay networks have proven an effective way of disseminating a file from a single source to a group of 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 [2], Slurpie [10], SplitStream [1] and Bullet [5]. 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 either from the server or from a peer who has previously downloaded it. More recently, a scheme based on network coding [3] has been suggested. Here, users download linear combinations of file parts rather than individual file parts.

Performance evaluation of such systems has typically been limited to comparing one system relative to another. Our results give the minimal time required to fully disseminate a file of M parts from a server to N end users. In the scheduling literature this completion time is referred to as *makespan*. We thus provide a lower bound which can be used as a performance benchmark for any P2P file dissemination system.

In [7] we analyzed a model of P2P file dissemination and found the minimal makespan when users have equal upload capacities. Now in this paper, we suppose user's upload capacities may differ. Other related work can be found in [6], [9] and [11].

2 The Uplink-Sharing Model

The uplink-sharing model of [7] is an abstract model focusing on the important features of P2P file dissemination. Each user can connect to every other user, i.e. the network topology is a complete graph. The server S has upload capacity C_S and the N users have upload capacities $C_1, \ldots, C_N$, measured in megabytes per second (MBps). We suppose 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. However, Lemma 2.1 below, taken from [7], shows that it suffices to restrict the server and all peers to carry out only a single upload at a time. We permit a user to download more than one file part simultaneously, but these must be from different sources. We ignore more complicated interactions and suppose that the upload capacities impose the only constraints on the rates at which file parts can be transfered between peers, which is a reasonable assumption if the underlying network is not overloaded. We also suppose that uploads and downloads do not constrain one another. We have shown in [7] 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 big. This is usually the case in practice. Lemma 2.1 and Lemma 2.2 from [7] dramatically simplify finding the minimal makespan. We also use them in this paper.

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

Lemma 2.2 *In the uplink-sharing model the minimal makespan is not increased by restricting uploads to start only at times that other uploads finish.*

3 Results

Essentially, we have an unusual precedence-constrained scheduling problem. It can be formulated as a so-called mixed integer linear program (MILP), by the following lemma, which says that time can be discretized suitably.

Lemma 3.1 *Suppose that all upload capacities are integer multiples of a common time unit. Then there exists a time τ and an optimal schedule such that all uploads start and finish at integer multiples of τ.*

The MILP is based on connection, exclusivity, continuity, stopping, source availability and link constraints. Some of these need to be linearized by means of introducing auxiliary variables and linearization constraints.

Theorem 3.2 *The general problem can be solved via a MILP formulation.*

MILPs are well understood theoretically and there exist efficient computational methods and program codes. As the numbers of variables and constraints grows exponentially in N and M, this approach is not practical when these are large, but we can also use our MILP formulation to obtain a bounded approximation to the solution.

We provide further insight into the solution by investigating a number of special cases.

[1]Statistical Laboratory, University of Cambridge, Cambridge CB2 0WB, UK

[2]Department of Statistics, University of Haifa, Mount Carmel 31905, ISRAEL

For very large M, the problem can be approximated by a fluid limit problem, in which the file is infinitely divisible. We obtain two surprisingly simple results.

Theorem 3.3 *The minimal makespan is*

$$T^* = \max\left\{\frac{1}{C_S}, \frac{N}{C_S + \sum_j C_j}\right\}.$$

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

In a still more general scenario each user i may have a file of size $F_i \geq 0$ to disseminate to all other users. As there is no longer a special distinguished server, we change notation so that there are $N \geq 2$ users in all. With $F = \sum_i F_i$ and $C = \sum_i C_i$, we find the following.

Theorem 3.4 *The minimal makespan is*

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

and this can still be achieved with a two-hop strategy.

The proof is straight forward for $N = 2$, since the minimal makespan is then $\max\{F_1/C_1, F_2/C_2\}$ and this is exactly the value of T^* in Theorem 3.4. Now suppose $N \geq 3$. It is easy to see that each of the $N + 1$ terms within the braces on the right hand side is a lower bound on the makespan, because each user has to upload each part of his file to at least one other user, which takes time F_i/C_i. Also observe that the total volume of files to be uploaded is $(N-1)F$ and the total available capacity is C. Thus the makespan can be no less than $(N-1)F/C$. It remains to show that a makespan of T^* can be achieved. In the proof of Theorem 3.4 we give a strategy with that property.
We conclude by discussing whether T^* is a good estimate for the minimal makespan if files are not infinitely divisible, but may only be divided into M parts. Often, $M >> \log(N)$ in practice (cf. [8]) and it turns out that this is sufficient for T^* to be a good approximation. Thus, we have provided an MILP formulation that is suitable for small values of M as well as a fluid limit solution that is suitable for typical and for large values of M.

4 Further Work

It would now be interesting to compare the makespan that can be achieved with various overlay networks directly to the minimal makespan. A mathematical analysis of the protocols is rarely tractable. However simulation or measurements can be used, for example, measurements for the BitTorrent protocol are reported in [4] and [8].
In practice, splitting the file and passing on extra information has an overhead cost. This is small (less than 0.1% according to [2]). Still, if there is one it will not be optimal to increase M beyond a certain value. This could be investigated.
In Internet applications users often connect for relatively short times. One could consider a dynamic setting in which peers arrive, and then leave when they have downloaded the file. Work in this direction, using a fluid model to study the steady-state performance, is pursued in [9] and there is other relevant work in [6] and [11]. Also of interest would be some consideration of free-riders. For example, BitTorrent implements a choking algorithm to limit free-riding.

References

[1] 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.

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

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

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

[5] 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.

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

[7] J. Mundinger and R. R. Weber. Scheduling of peer-to-peer file dissemination. Submitted for publication, 2005.

[8] 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.

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

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

[11] X. Yang and G. de Veciana. Performance of peer-to-peer networks: Service capacity and role of resource sharing policies. *Performance Evaluation, Special Issue on Performance Modeling and Evaluation of P2P Computing Systems*, 2005.