OPTIMAL STRATEGIES FOR EFFICIENT PEER-TO-PEER FILE SHARING

Mortada Mehyar, WeiHsin Gu, Steven H. Low, Michelle Effros, Tracey Ho

Engineering and Applied Science, California Institute of Technology
{morr, wgu, slow, effros, tho}@caltech.edu

ABSTRACT

We study a model for peer-to-peer file sharing. The goal is to distribute a file from a server to multiple peers. We assume the upload capacity of each peer is the only bottleneck. We examine the finish times of peers under different transmission strategies. Pareto optimality, min-max finish time, and optimal average finish time of the model are studied. We believe the results provide fundamental insights into practical peer-to-peer systems such as BitTorrent.

Index Terms— Peer-to-Peer networks, Mathematical Analysis/Optimization

I. INTRODUCTION

Peer-to-peer file sharing has become a significant part of internet traffic in recent years. According to web analysis firm CacheLogic, peer-to-peer traffic accounted for an astounding 60% of all internet traffic at the end of 2004.

Much research has gone into analyzing and understanding peer-to-peer file sharing systems such as BitTorrent [2], Slurpie [7], and Avalanche [3]. A fluid model for analyzing BitTorrent networks is proposed and studied in [6]. Issues related to the service capacity of peer-to-peer networks are studied in [8]. The work in this paper is motivated by the results in [4].

The model used in this work is introduced in [4] as the "uplink sharing model". A related model has been analyzed before in the context of message broadcasting among parallel processors [1]. One major difference between peer-to-peer systems and systems of parallel processors is that peer-to-peer systems are usually extremely heterogenous. Parallel processors, on the other hand, are usually assumed to be identical.

We extend the results in [4] and study different performance criteria. We also propose new directions for related research that we believe are promising for gaining insights on fundamental understanding of peer-to-peer file sharing systems.

II. PROBLEM SETUP

Consider a network of N peers where each peer's goal is to obtain the same content, a file of size F. The file is divided into P pieces of equal size to facilitate distribution. When a peer has received a piece completely, it can help distribute the piece by sending it to other peers. This method of file dissemination is used by file sharing systems such as BitTorrent [2], Slurpie [7], and Avalanche [3].

In addition to the N peers who initially have no file pieces, we assume there is a server that initially has the whole file. This server is called a seed node in BitTorrent. The upload capacity of each peer is assumed to be the only constraint, which is an assumption motivated by the fact that many peers have larger download capacities than upload capacities (e.g., DSL lines) on the Internet. We also assume that the overlay network is a complete graph, and therefore that there is no connectivity constraint.

We are interested in the time it takes for each peer to obtain the whole file. We denote by t_i the *finish time* of peer i, which is defined to be the earliest time at which peer i receives the whole file. The value of each t_i depends on the *strategy* with which the system allocates capacity and distributes data among peers.

When P, the number of pieces that make up the given file, is vary large, we can treat the data as a continuous flow. In this work, we focus on the case where P is infinity.

The notation for this model is as follows.

F: size of the file
N: total number of peers (not including the server)
C_s: upload capacity of the server
C_i: upload capacity of peer $i, i \in \{1, 2, ..., N\}$
$F_i(t)$: amount of file that peer i has at time t

Notice that by definition, we must have

$$0 \leq F_i(t) \leq F, \forall t. \tag{1}$$

Without loss of generality, we assume that the peer indices are ordered so that $C_i \leq C_j$ whenever $i > j$. We also define the total capacity of the system to be

$$C := C_s + \sum_{i=1}^{N} C_i.$$

We will also call the time F/C_s to be the *bottleneck time* of the system, since it is the least amount of time the server needs to upload the file F to any peer.

Notice that in this model, the analysis is not simply about allocating capacities. Peers can only upload they have already received. Also, an upload is only useful if the receiving peer has *not* already received the data. The analysis, therefore, has to take into account how different file segments are distributed, in addition to how capacities are allocated.

III. LAST FINISH TIME

We will call the amount of time for all peers to obtain the file the *last finish time*. Precisely $T_L := \max_i\{t_i\}$. This is called the "minimum makespan" in [4].

When the server and only the server has the file initially, it can be shown [5] that the minimal last finish time T_L^* is given by the following simple expression.

Theorem 1: [5, Theorem 3.3] The minimal last finish time is given by

$$T_L^* = \max\left\{\frac{F}{C_s}, \frac{NF}{C}\right\}. \tag{2}$$

1-4244-0728-1/07/$20.00 ©2007 IEEE

Mundinger et al. [4] discovered the following strategy that achieves the minimal last finish time as in (2), which we will denote by S_0.

(i) When

$$C_s \geq \frac{\sum_i C_i}{N-1}, \tag{3}$$

it is possible for the server to allocate to each peer i an upload rate of $\frac{C_i}{N-1}$. Peer i can therefore upload at the rate of

$$\frac{C_i}{N-1}$$

to *all* of its $N-1$ peers, without exceeding its capacity constraint C_i. The server's remaining upload capacity is

$$C_s - \frac{\sum_i C_i}{N-1}.$$

This will be shared equally among the N peers. Therefore, for any $i \in \{1, 2, \ldots, N\}$, the total capacity peer i receives equals

$$\frac{C_s - \frac{\sum_{j=1}^N C_j}{N-1}}{N} + \sum_{j=1}^{N} \frac{C_j}{N-1} = \frac{C}{N}. \tag{4}$$

(ii) When

$$C_s < \frac{\sum_i C_i}{N-1}, \tag{5}$$

the server can allocate to each peer i an upload rate of

$$\frac{C_i C_s}{\sum_{j=1}^N C_j}, \tag{6}$$

without exceeding its capacity constraint C_s. Peer i can therefore upload at the rate of

$$\frac{C_i C_s}{\sum_{j=1}^N C_j}$$

to *all* of its $N-1$ peers without exceeding its capacity constraint C_i, because by 5

$$(N-1)\frac{C_i C_s}{\sum_{j=1}^N C_j} < C_i. \tag{7}$$

Therefore the capacity each peer i receives is equal to

$$\sum_{i=1}^{N} \frac{C_i C_s}{\sum_{j=1}^N C_j} = C_s, \forall i \in \{1, 2, \ldots, N\}. \tag{8}$$

Unfortunately, minimizing the last finish time is not always good objective for peer-to-peer file sharing systems. Notice that the expression of T_L^* suggests that peers with small capacity can have an *arbitrarily large* impact on system performance. This is certainly not true in a BitTorrent network, for example. In fact, since strategy S_0 forces all peers to finish at the same time, the efficiency of the system is compromised. We skip the proof of following two lemmas.

Lemma 1: When $N = 2$, strategy S_0 is always Pareto-optimal. Strategy S_0 is not Pareto-optimal in general.

Lemma 2: When $N \geq 3$ and $T_L^* > F/C_s$, strategy S_0 is not Pareto-optimal.

IV. OTHER OPTIMALITY CRITERIA

IV-A. Average Finish Time

In the context of peer-to-peer file sharing, the last finish time may not be as important as other objectives such as *average finish time*, for example. The average finish time T_A is defined to be the average of all finish times

$$T_A := \frac{1}{N} \sum_{i=1}^{N} t_i. \tag{9}$$

A simple example illustrates why the average finish time may be a better measure of performance. Consider the special case where all peer capacities are 0. If server capacity C_s is split equally among all peers, every peer will finish at the same time

$$t_i = \frac{NF}{C_s}, \forall i \in \{1, 2, \ldots, N\}. \tag{10}$$

However if the downloads are done sequentially, the finish times can be

$$t_i = \frac{iF}{C_s}, \forall i \in \{1, 2, \ldots, N\}. \tag{11}$$

Therefore the average finish time in (11) becomes

$$\frac{(N+1)F}{2C_s}, \tag{12}$$

nearly half of the average value of the finish times in (10). The overall user experience is undoubtedly better in (11), and it can be shown that the finish times in (11) are actually *optimal* for average finish time.

IV-B. Min-Min Finish Times

As an alternative, we also consider the following "min-min" optimality criterion for the finish times, where each finish time t_i is sequentially minimized, in the order from fast to slow peers. We define

$$t_1^m \quad := \quad \min t_1 = \frac{F}{C_s}, \tag{13}$$

$$t_i^m \quad := \quad \min \left\{ t_i | t_j = t_j^m, \forall j < i \right\}. \tag{14}$$

In words, the min-min finish time t_i^m of peer i, is the minimal possible value of t_i subject to the constraints that the finish time of any peer j with an index $j < i$ is equal to $t_j = t_j^m$.

V. GENERAL PROPERTIES

We first mention without proving that within any period $(t_i, t_{i+1}]$, where no peer finishes, we can assume without loos of generality that no file segment is uploaded by more than two nodes. In other words, no file segment traverses more than two "hops" when it is sent.

Theorem 2: (Multiplicity Theorem) It is possible to let the first M peers finish at bottleneck time if and only if

$$C_s \leq \sum_{i=1}^{M} \frac{C_i}{M-1} + \sum_{i=M+1}^{N} \frac{C_i}{M}. \tag{15}$$

For any given system, we define the *multiplicity* of the system to be the largest M such that inequality (15) holds. In other words, the multiplicity is M if it is possible to finish the first M peers at bottleneck time, but not the first $M+1$ peers.

Note that the multiplicity of any system is at least 1, since it is always possible to finish peer 1 at bottleneck time. This fact can also be seen from the right hand side of (15), as it becomes infinity when $M = 1$, and therefore inequality (15) is true for any finite C_s.

One important consequence of the Multiplicity Theorem (Theorem 2) is that it suggests a natural decomposition of our model into cases according to the multiplicity.

VI. OPTIMAL AVERAGE FINISH TIME

Suppose the objective now is to minimize the average finish time. We next show that in many cases, the optimal average is achieved by the min-min finish times defined in Section IV-B.

First, we note that the optimal average finish time is easy to derive when the multiplicity M equals N, namely, when

$$C_s \leq \frac{\sum_{i=1}^{N} C_i}{N-1}. \tag{16}$$

Here since $t_i \geq F/C_s$ for all $i \in \{1, 2, ..., N\}$, the optimal average finish time must satisfy

$$T_A^* \geq \frac{F}{C_s}. \tag{17}$$

When $M = N$, strategy S_0 achieves

$$t_i = \frac{F}{C_s}$$

for all i. Therefore S_0 achieves min-min finish times and gives

$$T_A^* = \frac{F}{C_s}, \text{ if } M = N. \tag{18}$$

When the multiplicity is $N-1$, namely, when

$$\frac{\sum_{i=1}^{N} C_i}{N-1} < C_s \leq \frac{\sum_{i=1}^{N-1} C_i}{N-2} + \frac{C_N}{N-1}. \tag{19}$$

We can always write C_s in the form of

$$C_s = \lambda \left(\sum_{i=1}^{N-1} C_i \right) + \frac{C_N}{N-1}, \tag{20}$$

where λ is a constant such that

$$\frac{1}{N-1} < \lambda \leq \frac{1}{N-2}.$$

Consider the following strategy:

(i) The server uploads different file segments to each peer i, $i \in \{1, 2, \ldots, N-1\}$, at the rate of λC_i.
Peer i broadcasts the data it receives from the server to every other peer j. Peer i has remaining capacity $C_i - (N-2)\,\lambda C_i$, which it uses to upload information to peer N.

(ii) The server uploads to peer N at rate

$$\frac{C_N}{N-1}$$

, and peer N broadcasts the data it receives from the server to all other peers in the network. Note that the upload capacity of the server is saturated, and so is the upload capacity of peer N.

One can show that the above strategy results in finish times

$$t_i = \frac{F}{C_s}, \forall i \in \{1, 2, \ldots, N-1\}, \tag{21}$$

$$t_N = \frac{F}{C_s}\frac{NC_s - C_N}{C - C_N} \tag{22}$$

which minimizes the average finish time.

VI-A. Networks of Three Peers

When $N = 3$, the above analysis describes how to achieve the optimal average finish time provided $M > 1$. We next treat the case when $M = 1$. This is the case when

$$C_s \geq C_1 + C_2 + \frac{C_3}{2}. \tag{23}$$

Theorem 3 treats this case.

Theorem 3: When $N = 3$ and $M = 1$, the following strategy is optimal in average finish time:

(i) During time period $[0, t_1]$, the server sends different file segments to peers 1, 2, and 3 at rates $C_s - C_2 - r_3$, C_2, and r_3 respectively, where

$$r_3 := \frac{2C_s - C_2}{2C_s + 2C_1 + C_3} C_3. \tag{24}$$

Since

$$\frac{C_3}{2} \leq r_3 \leq \min\{C_3, C_s - C_1 - C_2\}, \tag{25}$$

the server can allocate rate r_3 to peer 3. Then peer 1 uploads to peer 2 at rate C_1, peer 2 uploads to peer 1 at rate C_2, peer 3 uploads to peer 1 at rate r_3, and to peer 2 at rate $C_3 - r_3$.

(ii) During time period $[t_1, t_2]$, peer 3 continues to upload to peer 2 the data it received from the server during $[0, t_1]$. The server and peer 1 each upload at full rate to peer 2, and peer 2 uploads at its full rate to peer 3.

(iii) During time period $[t_2, t_3]$, the server and peers 1 and 2 upload to peer 3 at a combined rate of $C_s + C_1 + C_2$, and finish the last peer.

Proof: Let the set of the peers $i \in \{1, 2\}$ be set A, then an upper bound of the total amount of data that can go into set A by time t_2 is

$$(C_s + C_1)t_2 + C_2 t_1 + \frac{C_3}{2} t_2. \tag{26}$$

The first term follows since the server and peer 1 can potentially upload data to set A at full capacity during $[0, t_2]$. The second term follows since peer 2 can only upload to A during $[0, t_1]$. During $[t_1, t_2]$, there is no destination in set A for peer 2 to upload to anymore. Since peer 3 can at most upload the same data to two peers, the net contribution from peer 3 to set A is at most

$$\min\{C_3 t_2, 2F_3(t_2)\} - F_3(t_2),$$

which is at most

$$\frac{C_3}{2} t_2.$$

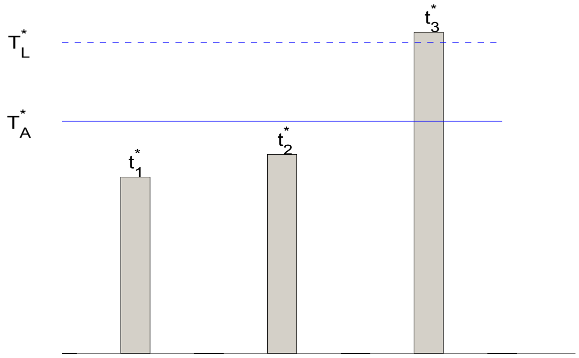

Fig. 1. An illustration of the finish times achieved by the strategy that minimizes average finish time. The parameters chosen are $C_s = 500$, $C_1 = 200$, $C_2 = 80$, and $C_3 = 70$. We also plot the optimal values T_A^* and T_L^*.
figure

Now, since peers 1 and 2 have both finished at time $t_1 = t_2$, at least $2F$ data must have been received. Therefore we have

$$C_s t_2 + C_1 t_2 + C_2 t_1 + \frac{C_3}{2} t_2 \geq 2F, \tag{27}$$

which is equivalent to

$$t_2 \geq \frac{2F - C_2 t_1}{C_s + C_1 + \frac{C_3}{2}}. \tag{28}$$

On the other hand, one can easily get

$$t_3 - t_2 \geq \frac{3F - C t_2}{C - C_3}. \tag{29}$$

Adding t_1 to both sides of (28), and adding $t_1 + t_2$ to both sides of (29) with $N = 3$ gives

$$t_1 \geq \frac{F}{C_s} \tag{30}$$

$$\sum_{i=1}^{2} t_i \geq \frac{2F + (C_s + C_1 + \frac{C_3}{2} - C_2)t_1}{C_s + C_1 + \frac{C_3}{2}} \tag{31}$$

$$\sum_{i=1}^{3} t_i \geq \frac{3F + C_3 t_1 + (C - 2C_3)(t_1 + t_2)}{C - C_3} \tag{32}$$

Plugging (30) and (31) into the right side of (32) gives a lower bound on $t_1 + t_2 + t_3$. Any strategy that achieves equality in each of the three inequalities (30-32), minimizes the average finish time. Since the strategy we propose achieves all three equalities, it minimizes average finish time. ■

Figure 1 illustrates numerically the finish times achieved by the strategy that minimizes average finish times. The optimal average finish time T_A^* is significantly better than the optimal last finish time T_L^*.

VII. SUMMARY AND CONCLUSION

We study a model for peer-to-peer file sharing with respect to different optimality criteria. We have derived general properties of the system, and analyzed special cases of system behavior. Intuitively, the results suggest that efficient peer-to-peer file sharing should have two components. One is that file segments should be spread out to as many peers as possible, in order to utilize every peer's upload capacity.

The other component is that fast peers should be favored over slow peers, but they should be not be favored to the exclusion of their peers. It seems that fast peers should receive more file segments earlier (but not all file segments), while at the same time the capacities of slow peers should also be utilized.

ACKNOWLEDGMENTS

We would like to thank Lachlan Andrew and Aliekber Gurel for helpful discussions.

VIII. 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] B. Cohen. Incentives build robustness in BitTorrent. *Proceedings of Workshop on Economics of Peer-to-Peer Systems*, 2003.

[3] C. Gkantsidis and P. Rodriguez. Network Coding for Large Scale Content Distribution. *Proceedings of IEEE Infocom*, Miami, 2005.

[4] J. Mundinger and R. Weber. Efficient File Dissemination using Peer-to-Peer Technology. Technical Report, Statistical Laboratory Research Reports 2004-01, Cambridge, January 2004

[5] J. Mundinger, R. R. Weber, and G. Weiss. Analysis of Peer-to-Peer File Dissemination amongst Users of Different Upload Capacities. *Performance Evaluation Review*, Performance 2005 Issue.

[6] D. Qiu and R. Srikant. Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. *Proceedings of ACM SIGCOMM*, Portland, 2004.

[7] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A Cooperative Bulk Data Transfer Protocol. *Proceedings of IEEE Infocom*, Hong Kong, 2004.

[8] X. Yang and G. de Veciana. Service capacity of peer to peer networks. *Proceedings of IEEE Infocom*, Hong Kong, 2004.