

ALTERNATIVE PERFORMANCE METRICS AND
OPTIMAL STRATEGIES FOR PEER-TO-PEER FILE
DISSEMINATION NETWORKS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Gary Matthew Ezovski

January 2009

© 2009 Gary Matthew Ezovski

ALL RIGHTS RESERVED

ABSTRACT

Peer-to-peer (P2P) file distribution is a scalable way to disseminate content to a wide audience. For a P2P network, one fundamental performance metric is the average time needed to deliver a certain file to all peers, which in general depends on the topology of the network and the scheduling of transmissions. Despite its apparent importance, how to minimize average finish time remains an open question even for a fully-connected network. This is mainly due to the analytical challenges that come with the combinatorial structures of the problem. In this thesis, by using the water-filling technique, we determine how each peer should use its capacity to sequentially minimize the file download times in an upload-constrained P2P network. Furthermore, it is shown that this scheduling also minimizes average finish time for the network. This optimality result not only provides fundamental insight to scheduling in such P2P systems, but also can serve as a benchmark to evaluate practical algorithms and illustrate the scalability of P2P networks. An overview of existing P2P systems is given, analytical challenges posed by the inclusion of download constraints are considered, and further areas for research in P2P optimality are discussed.

BIOGRAPHICAL SKETCH

Matthew Ezovski received the Bachelor of Science degree summa cum laude in electrical engineering / computer and systems engineering from Rensselaer Polytechnic Institute in May 2006. He is a recipient of the U.S. Department of Homeland Security Fellowship and was recently appointed to the IEEE-USA Committee on Communications and Information Policy. He has interned in the Sensor Development & Application Group at the National Institute of Standards and Technology, the Acoustic Signal Processing Branch at the U.S. Army Research Lab, the Washington Internships for Students of Engineering (WISE) program, and the Hardware Diagnostics and System Validation Engineering group at Mercury Computer Systems, Inc.

Matt is primarily interested in the areas of peer-to-peer networks and wireless communications, with particular interest in the area of file dissemination schemes and streaming media. Also of professional interest are issues in RFID technology, the worldwide deployment of the electronic passport, and general issues in science policy.

He also has been a clarinetist in the Cornell University Wind Ensemble and a copy editor for the *Cornell Daily Sun*.

ACKNOWLEDGEMENTS

Many people played a part in making my experience at Cornell an enjoyable one. If I neglected to include you here, my sincere apologies, but do know that I greatly appreciated all that you contributed over my 2.5 years in Ithaca.

First and foremost, I would like to thank my advisor, Prof. Kevin Tang, for all of his support and expertise along the way to this thesis. He certainly took a risk in allowing me to be his first student at Cornell, and I hope that I have lived up to expectations. I certainly wish him the best in his career at Cornell and wherever else it might take him, and am certain that he will do great things.

I'd also like to acknowledge and thank Prof. Lang Tong for guiding me through my first year at Cornell, and for helping me to understand and appreciate how research is done. His support has not gone unnoticed. I also must acknowledge Prof. Lachlan Andrew for all of his contributions to my research, and his generous gift of his time over much of the past 6 months.

Of course, I must note the incredibly diverse group of students who make up unit two of the School of Electrical and Computer Engineering. If you asked me six years ago where I thought I would be now, and with whom I would be working, I don't think I could have possibly imagined being with all of you, and it has been a privilege. Of particular note are my colleagues in Prof. Tang's group, Meng Wang and Enrique Mallada, who have been extremely supportive of my research and eccentricities. I'm confident that their sheer intelligence and focus will bring them great rewards.

The ACSP group was my original home in Cornell ECE, and so I must of course acknowledge Anima Anandkumar (without whom I'm not sure I would have survived), Ting He, Oliver Kosut, Parvathinathan Venkitasubramaniam, Abhishek Nadamani, Saswat Misra, Brandon Jones, and Aaron Lei. Stefan Geirhofer gets his own sentence in honor of being most responsible for keeping me sane. Also from within ECE, thanks

to Benjamin Kelly, Josh Gabet, Matt Gaubatz, Tony Kopa, Rob Karmazin, Jake Levy, and Jon Felbinger.

Some might say I shot myself in the foot through the time I spent with extracurricular endeavors. I always insist that I would not have made it through without them. Special thanks to the staff of the Cornell Daily Sun, especially Michael, Rebecca (both of you), Carlos, Julie, Noah, Jonny, and Olivia. As for the Wind Ensemble and Wind Symphony, well, there are just too many of you to list. Special thanks to Prof. Cynthia Johnston Turner and Matt Marsit.

And since this is now getting too long, on a personal note, I'd like to thank all of my friends and family who have helped me through the challenges of the past three years, especially my mother and father, Meg L., A.J., Stefan, Saswat, Matt M., Dan H., Adam, Ben L. (both of you), Cubby, Marie, Jake, Jon F., Jeff, Steph, Rob, Annie, Lindsey, Alan, Julie, Emily L., Christine, Colette, Cindi and Peter T., Liz, Jeannine, Abbie, Alex T., Brin, Steve J., Ryan S., Attila, Brooke, Katie W., and Shawn M.

Best wishes and thanks to all of you.

Matt

This research was performed under an appointment to the U.S. Department of Homeland Security (DHS) Scholarship and Fellowship Program, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE.

Portions reprinted, with permission, from [4] and [5]. (©2009 IEEE)

TABLE OF CONTENTS

Biographical Sketch	iii
Acknowledgements	iv
Table of Contents	vi
List of Figures	vii
1 Introduction	1
1.1 Overview	1
1.2 Related Work	2
1.3 Paper Organization	4
2 Problem Formulation	6
2.1 Model and Notation	6
2.2 Average Finish Time	8
2.3 Min-Min Times	12
3 Scheduling to Achieve Min-Min Times	15
3.1 Algorithm: Optimal Scheduling to Achieve Min-Min Times	17
3.2 Proof of Algorithm Optimality	20
4 Average Time Conclusions	28
4.1 Advantage of Heterogeneity	35
5 A Look at the Download-Constrained Case	37
6 Conclusion	40
Bibliography	42

LIST OF FIGURES

2.1	A diagram showing the constraints on communication between nodes in a 3-node plus server configuration.	7
2.2	Results showing improvement in the average finish time metric with small increase in last finish time	10
3.1	General water filling technique example diagram	16
3.2	Diagram of the water filling argument for $M = 1$	22
3.3	Diagram of the water filling argument for $1 < M < N$ and $C_0 > C_0^*$	25
4.1	Comparison of the variance of the capacity vector with the average finish time metric	35

CHAPTER 1

INTRODUCTION

1.1 Overview

The Internet makes an unimaginably large volume of information accessible around the globe. This information is of widely-varied type and modality, spanning the spectrum from very public to extremely secret, and including text, audio, and video. Given this diversity of data, and the diversity of the interests of those trying to use the Internet to share it, it should come as no surprise that not all data is treated in exactly the same way.

The method by which files are disseminated in today's Internet has less to do with the data itself and more to do with the buying power of the user looking to share the data. Large corporations with popular websites, such as Yahoo!, CNN, or Facebook, often employ *content distribution networks*, or CDNs, to increase their throughput across different localities and to ensure direct availability of materials [1]. Someone starting a small business or looking to share his latest home video, however, might not be able to afford the costs associated with establishing or contracting a CDN. Luckily, other options exist.

Peer-to-peer (P2P) networking utilities have allowed Internet users of all sizes and economic capacities to be able to distribute content without the need for high-capacity servers or CDNs. In the P2P model, each user, referred to as a *node*, acts both as an information sink for data it has not yet received and as an information source for information which it has received already. This contrasts with the traditional *client-server* model employed by CDNs and major web vendors.

The idea for today's P2P systems descended from earlier developments such as

Usenet and the Department of Defense's ARPANET systems, both of which were distributed, decentralized networks for file sharing. In the 1990's, as computer hardware advanced in speed and power, and digital audio and video became more pervasive and standardized, a greater wealth of data content was available to potentially be shared. In 1999, the Napster network became the first widely-used P2P system, leading to the proliferation of music downloads on an indexed network [11].

Recently, with the expansion of broadband end-user connections and the development of new P2P protocols, issues of performance and privacy have gained substantial interest. P2P applications are now among the most frequently used applications on the Internet and have often been observed to consume large fractions of available Internet bandwidth. In fact, studies [23], [24] have shown that upwards of 45% of Internet traffic can be attributed to P2P applications.

1.2 Related Work

Peer-to-peer (P2P) networking utilities have also generated a great deal of research activity in the last couple of years; see e.g., [6, 7, 8, 9, 16, 12, 24, 26, 28] and the references therein.

The fundamental advantage of peer-to-peer architectures compared with classic client-server architectures is their scalability. As every peer is both a client and a server at the same time, a P2P network can potentially distribute data to a large number of peers in a much shorter period of time. This paper considers a classical situation, in which a file is to be distributed as quickly as possible to a known set of peers. This can be used as a basic model for many scenarios such as distributing a software patch to an existing subscriber base. It is also a standard model used to illustrate the scalability of

P2P networks [13], in which one can calculate the amount of time needed to distribute a file of certain size to all peers under both P2P and client-server architectures. The calculation is typically done using the last finish time metric, which is defined to be the time when the last peer gets the complete file. Another natural fundamental metric is average finish time, which is the sum of finish times of all peers divided by the number of peers. However, minimizing it brings significantly more analytical challenges and this paper is devoted to the intermediate step of finding an explicit scheduling procedure to sequentially minimize the download times.

Many papers have explored performance of P2P networks [21, 12, 19, 29, 2, 27, 20, 17, 25], and several among these deal with optimal scheduling algorithms. For example, Mundinger et al. [20, 21] characterize the problem of file sharing in networks with heterogeneous upload capacities and discrete file divisions. They also explore initial results for cases where the file to be shared can be divided into infinitely small pieces. Another example is [12] which discusses optimal strategies for file distribution when multiple classes of service exist. Recently, Mehyar et al. [19] extended Mundinger's upload-constrained result and look at average finish time problems. They provide solutions for all cases in which the number of nodes is three or less, as well as solutions to a special class of larger cases. Building upon all this work while identifying new inductive structures and using new techniques such as water-filling, this paper provides a complete explicit algorithm to minimize average finish time with an arbitrary number of peers.

The main difficulty of the design of optimal file-distribution algorithms is the need to keep track of *data identity*. A node must receive a whole file, rather than just an amount of data equal to the file size, which could include much duplication. This complicates the problem of how a node should choose to send a piece of data from “who most needs

this *amount* of data?” to “who most needs this *particular piece* of data?” Ignoring this constraint significantly reduces the complexity of the problem [22] but produces unrealistic results. In general, how the overall network benefits from the decision to send a particular piece of data to a particular node depends on the optimality criterion, as well as the physical constraints of the nodes involved.

This paper is a step towards addressing the problem of designing explicit file dissemination scheduling algorithms which provably minimize average finish time. To overcome the aforementioned difficulty, our overall strategy is to use the intermediate step of introducing another optimality criterion—min-min times—which has an inherent inductive structure that facilitates algorithm design. This sub-problem is of independent interest when there is a chance that the network will be partitioned by network failures; by minimizing the time until another node has an entire copy of the file, this schedule improves the probability that all nodes will eventually be able to receive the complete file [15].

1.3 Paper Organization

The thesis is organized as follows. Chapter 2 presents the problem description and constraints, reviews the solution that achieves the optimal last finish time, and formulates the min-min and average finish time problems. In Chapter 3, a scheduling algorithm is presented, and it is shown that the algorithm solves the min-min time problem. This proof makes use of a novel *water-filling* approach, as adapted from classical information theory. We present a proof that the optimal min-min scheme achieves minimum average finish time in Chapter 4, and in Chapter 5 we provide some preliminary results and conjectures regarding the impact which download constraints would have on

the upload-constrained results. We conclude in Chapter 6 and discuss some possible interesting extensions.

CHAPTER 2
PROBLEM FORMULATION

2.1 Model and Notation

Consider a single node, referred to as the server, which needs to distribute a file of size $|F|$ to N peer nodes. The system is assumed to be churn-free, in that peers neither arrive nor leave. We further assume that there are no topological constraints; each node, including the server, can communicate with each other node with no bottlenecks other than the nodes' upload constraints. Finally, the file can be broken into infinitesimally small pieces; thus, there is no forwarding delay, and a node can immediately relay what it receives to another node.

This paper uses the following notation:

- $|F|$: size of the file
- $F_i(t)$: portion of the file that peer i has at time t
- $|F_i(t)|$: size of that portion
- N : total number of peer nodes (not including the server)
- C_0 : server upload capacity
- C_i : node i upload capacity $C_1 \geq C_2 \geq \dots \geq C_N$
- $C \triangleq C_0 + \sum_{i=1}^N C_i$: total system capacity
- $R_{ij}(t, t + \tau)$: data sent from node i to node j in the interval $(t, t + \tau)$.
- $r_{ij}(t) = \frac{d}{dt}|R_{ij}(0, t)|$: rate at which node i sends to node j at time t
- Finish time t_i for peer i : the smallest t with $|F_i(t)| = |F|$

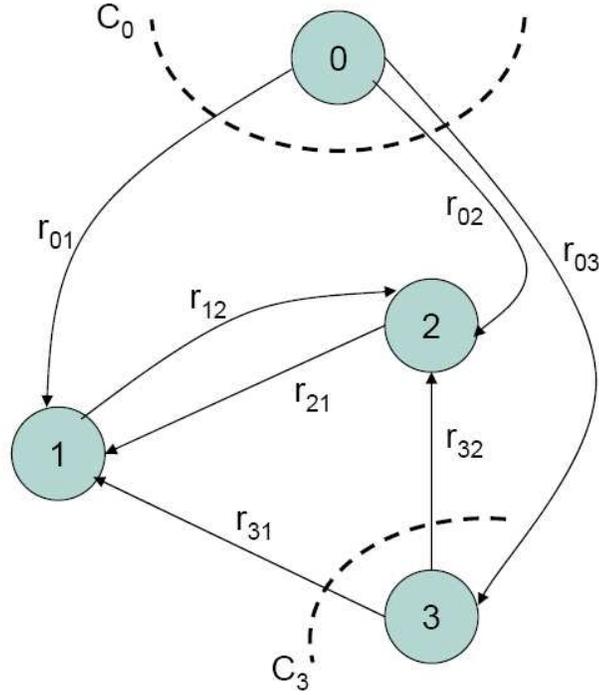


Figure 2.1: A diagram showing the constraints on communication between nodes in a 3-node plus server configuration. The dashed lines represent the sum-rate constraints $\sum_{j=0}^N r_{ij}(t) \leq C_i \forall i$ [5]. (©2009 IEEE)

- $|F|/C_0$ – bottleneck time: the time it takes for one node to directly receive the entire file from the server, and a lower bound on the time for all nodes to receive the file

We consider an upload-constrained scenario in which each node can receive information with unlimited data rate, but the sum rate of any uploads from each node must be no greater than that node’s given upload capacity. Mathematically,

$$\sum_{j=1}^N r_{ij}(t) \leq C_i \quad \forall i, t.$$

The “data identity” constraint can now be expressed as

- $R_{ij}(t, t + \tau) \subseteq F_j(t + \tau)$ (received data constraint; can only send data already received)
- $R_{ij}(t, t + \tau) \cap F_j(t) = \emptyset$ (only receive new data)
- $R_{ij}(t, t + \tau) \cap R_{kj}(t, t + \tau) = \emptyset \quad \forall i \neq k$ (only receive non-duplicate data)
- $r_{ii}(t) = 0$ (a node can't send data to itself)
- $F_j(t) = \bigcup_{i=0}^N R_{ij}(0, t)$, whence
- $\frac{d}{dt}|F_j(t)| = \sum_{i=0}^N r_{ij}(t) \quad \forall j, t.$

2.2 Average Finish Time

We first briefly review the problem of minimizing the last finish time (the time for all nodes in the network to receive the entire file). Clearly, this time, T_L^* , can't be less than $|F|/C_0$, which is the time it takes for the server to send the file to one recipient, or less than the time it would take to share the file with all nodes if every node in the network was fully utilized for all time, $N|F|/C$. Formally,

$$T_L^* \geq \max(|F|/C_0, N|F|/C) \quad (2.1)$$

Munding et al. [20] show that this lower bound is tight by looking at the following two possibilities.

Case 1 – Fast Server

When $C_0 \geq \sum_{i=1}^N C_i/(N - 1)$, each peer is assigned server capacity of rate $C_i/(N - 1)$, and each peer can then re-upload to the remaining $N - 1$ peers at rate $C_i/(N - 1)$. The excess capacity is shared equally. This results in each peer receiving total capacity C/N on the time interval $(0, T_L^*)$.

Case 2 – Slow Server

When $C_0 \leq \sum_{i=1}^N C_i / (N - 1)$, the server can allocate to each peer i an upload rate of

$$\frac{C_i C_0}{\sum_{j=1}^N C_j}$$

which does not exceed that peer's upload capacity. Each node can forward on what it receives to every other peer; thus, each peer effectively receives at rate C_0 from the server.

Forcing all the nodes to finish receiving the file at T_L^* might artificially limit the performance of the network by other metrics; for example, allowing small increases in $T_L > T_L^*$, we can potentially substantially decrease the average finish time, T_A , and thus improve the overall performance of the network. This is illustrated with the following simple numerical example.

Example 1: *Potential improvement over minimizing last finish time.*

Let $N = 4$, with $C_0 = 12$, $C_1 = 6$, $C_2 = 4$, $C_3 = 2$, $C_4 = 1$, and $|F| = 144$. We calculate the optimal last finish time T_L^* and the optimal average finish time, T_A . The results are summarized in Fig. 2.2. By allowing a very small upward shift in finish time t_4 , substantial improvements in other finish times can be achieved. For example, with the selected set of upload capacities and specified file size, an average finish time decrease of 28.9% corresponds to a 0.91% increase in last finish time.

It is now clear that the average finish time is an important performance metric. Formally, we have

$$T_A = \frac{\sum_{i=1}^N t_i}{N}. \tag{2.2}$$

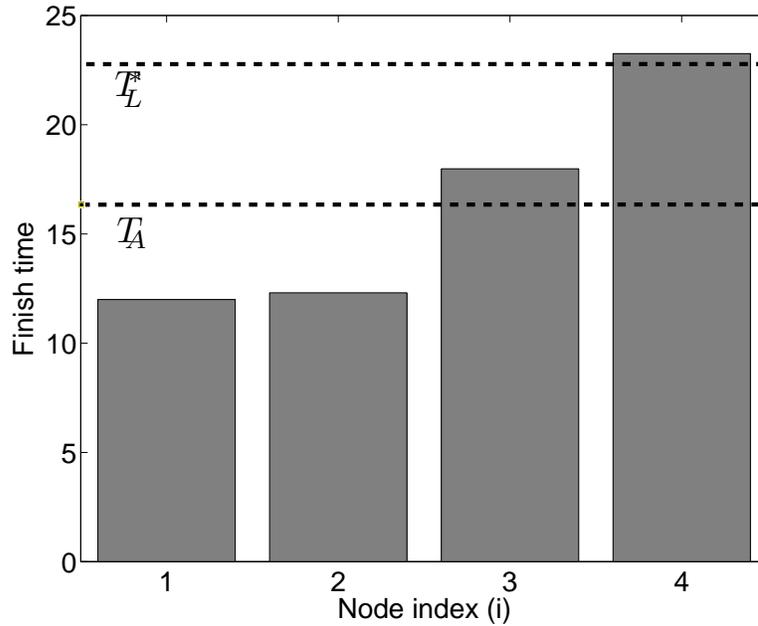


Figure 2.2: Results for the $N = 4$ case, with $C_0 = 12$, $C_1 = 6$, $C_2 = 4$, $C_3 = 2$, $C_4 = 1$, and $|F| = 144$. T_A is the associated average finish time, and T_L^* is the optimal last finish time [5]. (©2009 IEEE)

In general, to minimize the average finish time, we want to maximize the rate at which information is exchanged in the network for all times, and attempt to minimize the finish times of nodes with high capacity as quickly as possible. However, due to the combinatorial structure of the problem and especially the data identity constraint, it is hard even to write down a tractable optimization problem for general case. The following example illustrates the difficulty of writing a linear program to solve this problem for a very simple 2-peer network.

Example 2: *Direct minimizing average finish time for a two-peer network.*

Considering the 2-peer case, we can set up a linear program which optimizes the average finish time by adjusting the sizes of the blocks of data the nodes send to each

other in each time interval within the constraints of the problem.

$$\begin{aligned}
& \min_{R_{01}, R_{02}, R_{12}} && t_1 + t_2 \\
\text{subject to} &&& t_1 = |R_{01}(0, t_1)| / (\lambda C_0) \\
&&& t_2 = t_1 + (|R_{01}(0, t_1)| - |R_{12}(0, t_1)| \\
&&& \quad - \frac{|R_{01}(0, t_1) \cap R_{02}(0, t_1)|}{(C_1 + C_0)}) \\
&&& \lambda = |R_{01}(0, t_1)| / (|R_{01}(0, t_1)| + |R_{02}(0, t_1)|) \\
&&& |R_{01}(0, t_1) \cup R_{02}(0, t_1)| = |F| \\
&&& |R_{21}(0, t_1)| \leq C_2 t_1 \\
&&& |R_{12}(0, t_1)| \leq C_1 t_1 \\
&&& |R_{01}(0, t_1)| + |R_{02}(0, t_1)| = C_0 t_1 \\
&&& |R_{21}(0, t_1)| = |R_{02}(0, t_1) \setminus R_{01}(0, t_1)| \\
&&& |R_{12}(0, t_1)| \leq |R_{01}(0, t_1)|
\end{aligned}$$

Here the data identity constraint forces us to keep track of the sizes of many distinct pieces of data even when $N = 2$ (the last six constraints in the above optimization). In general, similar optimizations can be written for larger N , but the number of variables and constraints grows exponentially with the size of the problem. This difficulty motivates us to look for inductive structures which allow us to avoid such a construction. The min-min times structure that will be introduced in section 2.3 serves this role.

2.3 Min-Min Times

Min-min times minimize the individual node finish times. Besides the relation to the optimal average finish time, the structure is also of independent interest, since minimizing the completion times of early flows improves the network's robustness to disconnection [15].

Formally, Let t_i^s be the finish time of peer i under rate scheme s .

- Let S_1 be the set of schemes which minimize time t_1 .
- Let S_{i+1} be the set of schemes which minimize the $i + 1$ st finish time, given that $S_{i+1} \in S_i$.

A scheme in $s \in S_N$ is said to achieve *min-min times*, and the times t_i^s are called the min-min times.

The inductive structure imposed by sequential minimization allows us to find an explicit schedule to achieve min-min times. This will be shown in Chapter 3. Before delving into our main results, however, we introduce the useful concept of *multiplicity* [19], which will be used to classify problems. Define multiplicity, M , as the maximum number of nodes which can receive a file with size $|F|$ in bottleneck time $|F|/C_0$.

Lemma 1 *Let M be the largest value of K such that there exists a schedule with*

$$F_i\left(\frac{|F|}{C_0}\right) = F, \quad \forall i \leq K.$$

Then M is the largest integer such that

$$C_0 \leq \sum_{i=1}^M \frac{C_i}{M-1} + \sum_{M+1}^N \frac{C_i}{M}. \quad (2.3)$$

Proof of Lemma 1 (from [18]) *Let nodes $1, \dots, M$ make up the set A_1 , and nodes $M + 1, \dots, N$ make up the set A_2 . Note that the server and peers in the set A_1 can upload at a rate of at most*

$$C_0 + \sum_{i=1}^M C_i$$

, and that the server can route through peer $j \in A_2$ at a cost. If data is routed through j to all nodes in set A_1 , this results in a copy of the data being left on node j and a net contribution factor of $M - 1$. C_j/M is thus the maximum server capacity allocation which can be made to node j if M peers are to receive from it in bottleneck time. The maximum contribution of the nodes in A_2 is therefore

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

In order for all M peers in A_1 to receive the entire file F in bottleneck time, M copies of F must be sent to the set A_1 . Based on the maximum multiplicative contribution of A_2 ,

$$M|F| \leq (C_0 + \sum_{i=1}^M C_i + \sum_{i=M+1}^N \frac{M-1}{M} C_i) \frac{|F|}{C_0}$$

which simplifies to

$$C_0 \leq \sum_{i=1}^M \frac{C_i}{M-1} + \sum_{i=M+1}^N \frac{C_i}{M}.$$

The A_1, A_2 partition is optimal if ordered by capacities in that it maximizes the possible contribution to the set A_1 for a given number of nodes M in that set.

Let $0 < \lambda \leq 1$. If the server sends to each node in A_1 at rate

$$\lambda \frac{C_i}{M-1}$$

and each node in A_2 at rate

$$\lambda \frac{C_i}{M},$$

and each node forwards what it receives from the server to each other node in A_1 , then each node in A_1 receives a stream from the server at rate C_0 and will finish in bottleneck time. ■

The concept of multiplicity will be useful for separating sets of upload capacities into classes with distinct solutions.

CHAPTER 3

SCHEDULING TO ACHIEVE MIN-MIN TIMES

For the class of problems with multiplicity $M = N$, all nodes can finish by $|F|/C_0$ using the schedule reviewed in section 2.2. We now study optimal schedules for the remaining cases ($M < N$).

The main difficulty in achieving min-min times is determining how to use the extra capacities of some peers when trying to minimize t_i . It will be shown that they only need to focus on minimizing t_{i+1} ; in other words, looking more than one step ahead is generally not useful. Another difficulty is determining how to schedule peers given that they all have different capacities C_j . A “water filling” technique will be used to decide optimal scheduling for all peers. The potential contributions of finished nodes to the next finishing node can be thought of as “water”, and the data scheduled to be shared by other nodes forms an uneven floor.

This technique is illustrated in Fig. 3.1(a). During the interval (t_{i-1}, t_i) , the j th column has width C_j , and area $F_j(t_1) \setminus F_i(t_{i-1})$. Thus, the depth is the minimum time it would take for node j to upload all of the data it could to node i .

Note that the sets $F_j(t_1) \setminus F_i(t_{i-1})$ are disjoint for $j > i$. (This will be guaranteed by our scheduling algorithm.) Thus, the region in columns $j > i$ is exactly the data which must be transmitted to node i in the interval (t_{i-1}, t_i) , and the question is who should transmit what to minimize this interval. If the server and completed nodes did not send any further data to node i , the maximum depth is the minimum possible value of $t_i - t_{i-1}$ (column N in Fig. 3.1). The optimal way is to let nodes $0 \leq j < i$ send the shaded data in Fig. 3.1(b), equalizing the finish times $t_i - t_{i-1} = |F_j(t_1) \setminus F_i(t_{i-1})|/C_j$.

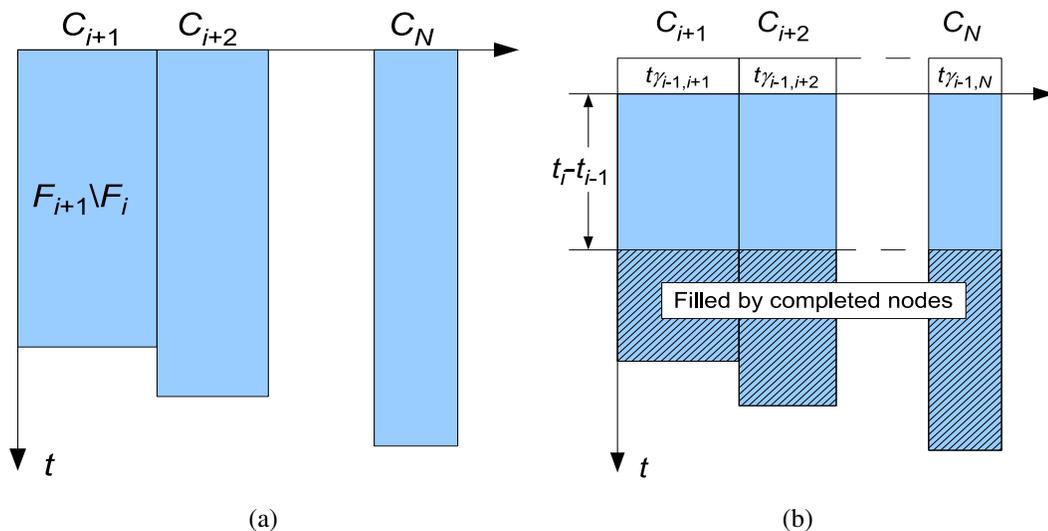


Figure 3.1: Water filling. The width of column j is capacity C_j , and the depth is the time to transmit $F_j(t_1) \setminus F_i(t_{i-1})$ at rate C_j . In (a), node N takes longer to transmit its information. In (b), the server has *water-filled*, decreasing the time for all to complete transmission to node i , and allowing full utilization for the interval. The *helium-filling* by $(t_i - t_{i-1})\gamma_{ij}$ in interval (t_{i-1}, t_i) reduces the heights of all columns equally [5]. (©2009 IEEE)

The only remaining question is what node i should do when others are uploading data to it. Due to the “data identity” constraint, $r_{ii}(t) = 0$, and therefore it cannot transfer data to itself. The optimal way is to use node i ’s capacity to send data to $i + 1$. The specific data to be sent will be determined by “helium-filling” for the following time interval, (t_i, t_{i+1}) as follows: Data U_{ij} , sent at rate γ_{ij} , is chosen such that it would have been in column j at time t_i had it not been sent to node $i + 1$ in interval (t_{i-1}, t_i) , but it instead “comes off the top” of the columns, in proportion to their capacities (Fig. 3.1(b)). (Note that U_{i0} corresponds to data which would have been sent by the server on the interval (t_i, t_{i+1}) , and thus is represented by “water,” but is instead sent by node i on the previous interval.) In later proofs, we will provide specialized water-filling figures for different cases (Figures 3.2 and 3.3).

The actual scheduling algorithm is stated in section 3.1. It uses C_0^* , which is an upper bound on the range of C_0 for a particular given multiplicity M for which exactly one set of optimal values $F_i(|F|/C_0)$, $\forall i > M + 1$, is able to achieve first $M + 1$ min-min times. Formally, it is the solution to

$$\begin{aligned} & \frac{M(C_0^* - \frac{C_{M+1}}{M} - \sum_{i=1}^M \frac{C_i}{M-1}) + C_0^*}{C_0^* + \sum_{i=1}^M C_i} \\ &= \frac{(M+1)(C_0^* - \frac{C_{M+1}}{M} - \sum_{i=1}^M \frac{C_i}{M-1})}{\sum_{i=M+2}^N C_i}. \end{aligned} \quad (3.1)$$

When $C_0 > C_0^*$, there could be multiple sets of $F_i(|F|/C_0)$, $\forall i > M + 1$ that all achieve the first $M + 1$ min-min times. Then the algorithm also uses the following linear program to select the only set of $F_i(|F|/C_0)$ values which allows all min-min times to be achieved.

$$\max \sum_{i=M+2}^N (N-i)\lambda_i \quad (3.2)$$

s.t.

$$\begin{aligned} & \frac{C_i}{M+1} < \lambda_i \leq \frac{C_i}{M} \quad \forall i \geq M+2 \\ & \sum_{i=M+2}^N \lambda_i = C_0 - \frac{C_{M+1}}{M} - \sum_{i=1}^M \frac{C_i}{M-1} \\ & \frac{(M+1)\lambda_i - C_i}{C_i} \geq \\ & \frac{\frac{1}{M-1} \sum_{i=1}^M C_i + (M+1) \sum_{i=M+2}^N \lambda_i - \sum_{i=M+2}^N C_i}{C - C_{M+1}} \end{aligned}$$

3.1 Algorithm: Optimal Scheduling to Achieve Min-Min Times

If $M = 1$,

- Let $r_{0j} = \lambda_j$, $r_{j1} = \min(\lambda_j, c_j)$, $r_{j2} = c_j - \min(\lambda_j, c_j)$, where λ_j satisfy

$$\sum_{j=1}^N \lambda_j = C_0 \quad \lambda_2 = C_2 \quad \frac{2\lambda_j}{C_j} = \frac{\lambda_1 + C_0}{C_0 + C_1}, \forall j > 2 \quad (3.3)$$

Break file F into N partitions, $\Lambda_1, \dots, \Lambda_N$, where $|\Lambda_j| = \lambda_j \frac{F}{C_0}$. Set $R_{0j}(0, \frac{|F|}{C_0}) = \Lambda_j$.

Set marker $G_{1,j}$ within fragment Λ_j at and of subset $\Lambda_i(0, r_{j2} \frac{|F|}{C_0})$. Set $R_{j2} = \Lambda_j(0, G_{1,j})$.

- On (t_{i-1}, t_i) , $2 \leq i < N$:

$r_{ji}(t) = C_j \quad \forall j \neq i$. Set

$$t_i - t_{i-1} = \frac{\sum_{k \neq i} |\Lambda_k| - G_{i-1,k}}{\sum_{k \neq i} C_k}.$$

Set $R_{ji} = \Lambda_j(G_{i-1,j}, G_{i-1,j} + C_j(t_i - t_{i-1}))$ for all $j \neq i, 0$, and $R_{0i} = \bigcup_{j \neq i, 0} \Lambda_j(G_{i-1,j} + C_j(t_i - t_{i-1}), |\Lambda_j|)$.

Set $r_{i,i+1}(t) = C_i$. $R_{i,i+1} = B + \bigcup_{j \neq 0, i+1} \Lambda_j(0, \gamma_{ij}(t_i - t_{i-1}))$, where γ_{ij} is selected according to

$$\gamma_{ij} = \frac{C_j}{\left(\sum_{k=0, k \neq i+1}^N C_k\right)} C_i.$$

and B is a subset of R_{0i} for the time interval (t_i, t_{i+1}) of size $\gamma_{i0}(t_i - t_{i-1})$ which will be sent on the current time interval instead of the next time interval.

Set $G_{i,j} = \gamma_{ij}(t_i - t_{i-1})$ for all j .

Else

- If $M = N - 1$ or $C_0 \leq C_0^*$, given by (3.1),

Then let λ_i solve

$$\sum_{i=1}^N \lambda_i = C_0$$

$$\frac{\lambda_i}{C_i} = \begin{cases} \lambda_1/C_1 & \text{if } i \leq M \\ 1/M & \text{if } i = M + 1 \\ \lambda_{M+2}/C_{M+2} & \text{if } i \geq M + 2 \end{cases} \quad (3.4)$$

$$\frac{\lambda_{M+2}}{C_{M+2}} = \frac{M + \sum_{i=1}^M \lambda_i + C_0}{(M+1) \sum_{i=0}^M C_i}$$

Else let $\lambda_i = C_i/(M-1)$, $\forall i \leq M$, and let λ_i for $i \geq M+2$ satisfy the LP (3.2).

Endif

- On $(0, t_M)$:

$$r_{0i} = \lambda_i, \forall i; r_{ij}(t) = \lambda_i \text{ for } j \leq M, j \neq i; r_{ij}(t) = 0 \text{ for } j > M+1; r_{i,M+1}(t) = C_i - \sum_{j \neq M+1} r_{ij}(t).$$

Break file F into N partitions, $\Lambda_1, \dots, \Lambda_N$, where $|\Lambda_j| = \lambda_j \frac{F}{C_0}$. Set $R_{0j}(0, \frac{|F|}{C_0}) = \Lambda_j$.

Set marker $G_{M,j}$ within fragment Λ_j at and of subset $\Lambda_i(0, r_{j,M+1} \frac{|F|}{C_0})$. Set $R_{j,M+1} = \Lambda_j(0, G_{M,j})$.

- On (t_{i-1}, t_i) for $M+1 \leq i < N$:

$$r_{ji}(t) = C_j \quad \forall j \neq i. \text{ Set}$$

$$t_i - t_{i-1} = \frac{\sum_{k \neq i} |\Lambda_k| - G_{i-1,k}}{\sum_{k \neq i} C_k}.$$

Set $R_{ji} = \Lambda_j(G_{i-1,j}, G_{i-1,j} + C_j(t_i - t_{i-1}))$ for all $j \neq i, 0$, and $R_{0i} = \bigcup_{j \neq i, 0} \Lambda_j(G_{i-1,j} + C_j(t_i - t_{i-1}), |\Lambda_j|)$.

Set $r_{i,i+1}(t) = C_i$. $R_{i,i+1} = B + \bigcup_{j \neq 0, i+1} \Lambda_j(0, \gamma_{ij}(t_i - t_{i-1}))$, where γ_{ij} is selected according to

$$\gamma_{ij} = \frac{C_j}{\left(\sum_{k=0, k \neq i+1}^N C_k\right)} C_i.$$

and B is a subset of R_{0i} for the time interval (t_i, t_{i+1}) of size $\gamma_{i0}(t_i - t_{i-1})$ which will be sent on the current time interval instead of the next time interval.

Set $G_{i,j} = \gamma_{ij}(t_i - t_{i-1})$ for all j .

EndIf

On (t_{N-1}, t_N) , $r_{iN}(t) = C_i$ for $i < N$, and $r_{ik}(t) = 0 \quad \forall k$, sending remaining pieces of file.

3.2 Proof of Algorithm Optimality

The following theorem characterizes the scheduling algorithm in section 3.1.

Theorem 1 *The algorithm in section 3.1 achieves min-min times.*

Proof of Theorem 1 • *If $M = 1$:*

Note first that the algorithm is feasible. In particular, until time t_i , all nodes $j > i$ have disjoint data, while nodes $j < i$ have all data. Similarly, $U_{i,j}$, the subset of Λ_j sent by node i to node $i + 1$ on (t_{i-1}, t_i) , can be forwarded by i as it is received from node j , since $\gamma_{i,j} \leq C_j$, allowing the three claimed conditions to be satisfied.

It remains to establish optimality. Let \underline{t}_i denote the min-min finish time of node i . The proof of optimality first establishes lower bounds on \underline{t}_1 and \underline{t}_2 , and shows that Algorithm 3.1 achieves those times, and that the λ_i are the unique values which can achieve that. It then inductively shows that subsequent times are minimized.

Let $C' = \sum_{i=3}^N C_i$. This can be thought of as the capacity of a “virtual node” consisting of nodes $3, \dots, N$. As in [19], the amount of information that can go into nodes 1 and 2 on $(0, t_2)$ is bounded above as

$$F_1(t_2) + F_2(t_2) \leq (C_0 + C_1)t_2 + C_2t_1 + \frac{C'}{2}t_2. \quad (3.5)$$

The first terms shows that the server and node 1 can contribute on the whole time interval. The second term reflects node 2’s transmission to node 1 on $(0, t_1)$; on (t_1, t_2) , it cannot contribute, since it cannot upload to itself, and on (t_1, t_2) node 1

has already received the whole file. The term $t_2 C' / 2$ arises as follows. Node $i \geq 3$ can send information which it has received up to time t_2 to both nodes 1 and 2, but it cannot exceed its own upload capacity, and cannot upload to t_1 data which it does not have until t_1 . Thus, its uploads to 1 and 2 are bounded above by $\min \{C_i t_2, F_i(t_1) + F_i(t_2)\}$. However, the data obtained by node i from the server comes at the expense of data that the server could have sent to node 1 or 2 directly, giving a net contribution of

$$\min \{C_i t_2, F_i(t_1) + F_i(t_2)\} - F_i(t_2). \quad (3.6)$$

Note that

$$\min \{C_i t_2, F_i(t_1) + F_i(t_2)\} \leq \frac{C_i t_2 + 2F_i(t_2)}{2} \quad (3.7)$$

with equality only if

$$2F_i(t_1) = 2F_i(t_2) = C_i t_2. \quad (3.8)$$

Substituting (3.7) into (3.6) and summing over $i \geq 3$ gives $C' t_2 / 2$, establishing (3.5).

A lower bound on t_2 results from substituting $F_1(t_2) + F_2(t_2) = 2F$ into (3.5), and substituting the known value $t_1 = |F|/C_0$, giving

$$t_2 \geq \frac{2|F| - C_2|F|/C_0}{C_0 + C_1 + C'/2}. \quad (3.9)$$

This is achieved by the algorithm.

To see that the choice of λ_i is the only one which achieves t_2 , note that (3.8) is a necessary condition for all $i \geq 3$. Dividing by $C_i t_1$ and substituting $\lambda_i = |F_i(t_1)|/t_1$ gives

$$\frac{2\lambda_i}{C_i} = \frac{t_2}{t_1} \quad (3.10)$$

for all $i \geq 3$. Similarly, the data known only to node 1 and the server at t_1 , of which there is an amount $(\lambda_1 - C_1)t_1$, must also be delivered at rate $C_1 + C_0$ in

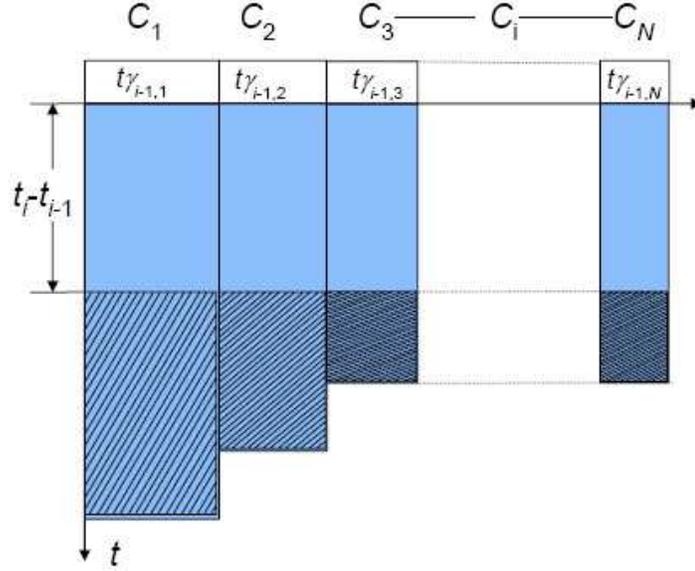


Figure 3.2: A visual depiction of the water filling argument for the case when $M = 1$. Note that the bottoms of columns $M + 2, \dots, N$ are level [5].
(©2009 IEEE)

time $t_2 - t_1$. Dividing by t_1 and adding 1 gives

$$\frac{\lambda_1 + C_0}{C_0 + C_1} = \frac{t_2}{t_1}. \quad (3.11)$$

Combining (3.10) and (3.11) shows that λ_i , $i > 2$, must satisfy (3.3) to achieve t_2 . Thus, Algorithm 3.1 achieves t_1 and t_2 , and (3.3) are necessary for any scheme which does.

Given that (3.3) must hold in order to achieve t_1 and t_2 , it can be shown by induction on i that: (a) node i receives no data in the interval (t_1, t_{i-2}) , and (b) t_i is tightly bounded below by

$$t_i \geq \frac{|F| - \lambda_i t_1 - C_{i-1}(t_{i-1} - t_{i-2})}{C - C_i} + t_{i-1}. \quad (3.12)$$

The term $\lambda_i t_1$ is the amount of data received by node i from the server during the first interval, $(0, t_1)$, and the term $C_{i-1}(t_{i-1} - t_{i-2})$ is the data received from node

$i - 1$ in the interval $(\underline{t}_{i-2} - \underline{t}_{i-1})$. Minimizing the latter term requires that node $i + 1$ receives no data in the interval (t_1, t_{i-1}) . The algorithm satisfies that and hence establishes the inductive step.

- If $M > 1$:

The proof begins by establishing conditions for appropriate λ values. It then finds the exact values of λ and min-min times $\underline{t}_1, \dots, \underline{t}_{M+1}$, and applies the water/helium-filling concept to establish all remaining min-min times.

In order to achieve minimum $\underline{t}_1 \dots \underline{t}_M$, each node must relay whatever it receives from the server on $(0, \underline{t}_M)$ to nodes $i \in \{1, \dots, M\}$. Thus, an upper bound on what each node can receive from the server on $(0, t_M)$ is

$$\lambda_i \leq \frac{C_i}{M-1} \quad \forall i \leq M \quad (3.13)$$

$$\lambda_i \leq \frac{C_i}{M} \quad \forall i > M \quad (3.14)$$

Since the algorithm keeps λ_i values in these ranges, and relays all server streams to nodes $\{1, \dots, M\}$, times $\underline{t}_1, \dots, \underline{t}_M = |F|/C_0$ are minimized.

To establish a lower bound on \underline{t}_{M+1} , consider first how much data node $M + 1$ can receive on $(0, t_M)$, from the server, nodes $\{1, \dots, M\}$, itself, and nodes $\{M + 2, \dots, N\}$:

$$\begin{aligned} |R_{0,M+1}(0, t_M)| &= \lambda_{M+1} t_M \\ \left| \bigcup_{i=1}^M R_{i,M+1}(0, t_M) \right| &\leq \left(\sum_{i=1}^M C_i - (M-1) \sum_{i=1}^M \lambda_i \right) t_M \\ |R_{M+1,M+1}(0, t_M)| &= 0 \\ \left| \bigcup_{i=M+2}^N R_{i,M+1}(0, t_M) \right| &\leq \left(\sum_{i=M+2}^N C_i - M \sum_{i=M+2}^N \lambda_i \right) t_M \end{aligned}$$

On (t_M, t_{M+1}) , each node $i \in \{0, 1, \dots, M\}$ could send to $M + 1$ with rate $r_{i,M+1}(t) =$

C_i , giving

$$\left| \bigcup_{i=0}^M R_{i,M+1}(t_M, t_{M+1}) \right| \leq (t_{M+1} - t_M) \sum_{i=0}^M C_i. \quad (3.15)$$

The contribution $\bigcup_{i=M+2}^N R_{i,M+1}(t_M, t_{M+1})$ of nodes $\{M+2, \dots, N\}$ is limited both by their sum upload capacity, $\sum_{i=M+2}^N C_i$, and by the amount of information they received on $(0, t_M)$. Thus

$$\left| \bigcup_{i=M+2}^N R_{i,M+1}(t_M, t_{M+1}) \right| \leq \min \left(\sum_{i=M+2}^N C_i (t_{M+1} - t_M), \sum_{i=M+2}^N \lambda_i t_1 - \left[\sum_{i=M+2}^N C_i - \sum_{i=M+2}^N \lambda_i M \right] t_M \right). \quad (3.16)$$

These combine to form the upper bound on the amount of information which can be received by node $M+1$ by time t_{M+1} .

$$\begin{aligned} |F_{M+1}(t_{M+1})| &\leq \left(\sum_{i=1}^M C_i - (M-1) \sum_{i=1}^M \lambda_i \right) t_M - M \sum_{i=M+2}^N \lambda_i t_M + \lambda_{M+1} t_M \quad (3.17) \\ &\quad + (t_{M+1} - t_M) \left(C_0 + \sum_{i=1}^M C_i \right) + \min \left(\sum_{i=M+2}^N C_i t_{M+1}, \sum_{i=M+2}^N (M+1) \lambda_i t_1 \right) \end{aligned}$$

Also note that by definition, $F_{M+1}(t_{M+1}) = F$.

Considering each term of the min in (3.17) separately, and solving for t_{M+1} yields two lower bounds on t_{M+1} in terms of $\sum_{i=1}^M \lambda_i$, λ_{M+1} , and $\sum_{i=M+2}^N \lambda_i$.

When $\sum_{i=M+2}^N C_i t_{M+1} \leq \sum_{i=M+2}^N (M+1) \lambda_i t_1$,

$$\begin{aligned} \underline{t}_{M+1} (C - C_{M+1}) &\geq \underline{t}_M (M-1) \sum_{i=1}^M \lambda_i - \underline{t}_M \lambda_{M+1} \quad (3.18) \\ &\quad + \underline{t}_M C_0 + \underline{t}_M M \sum_{i=M+2}^N \lambda_i + |F| \end{aligned}$$

and in the converse case

$$\begin{aligned} \underline{t}_{M+1} \sum_{i=0}^M C_i &\geq \underline{t}_M (M-1) \sum_{i=1}^M \lambda_i - \underline{t}_M \lambda_{M+1} \quad (3.19) \\ &\quad + \underline{t}_M C_0 - \underline{t}_M \sum_{i=M+2}^N \lambda_i + |F|. \end{aligned}$$

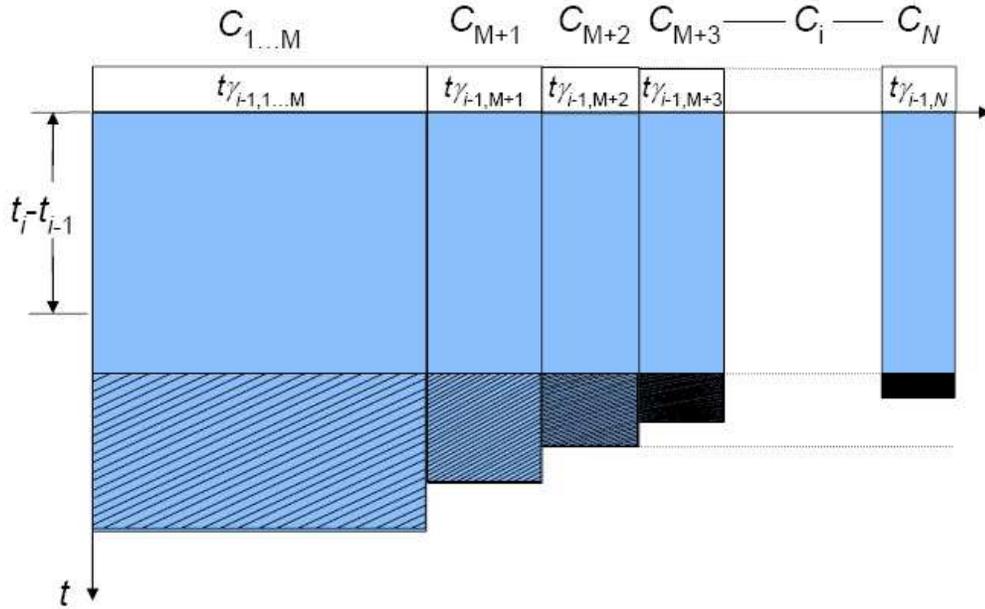


Figure 3.3: A visual depiction of the waterfilling argument for the case when $1 < M < N$ and $C_0 > C_0^*$. Note the tiered structure of the columns for $i > M$ [5]. (©2009 IEEE)

Note that in both cases, the lower bound is decreasing in λ_{M+1} , and so is minimized by maximizing λ_{M+1} by setting

$$\lambda_{M+1} = \frac{C_{M+1}}{M}. \quad (3.20)$$

Since the bound given by (3.18) is increasing in $\sum_{i=M+2}^N \lambda_i$ and that given by (3.19) is decreasing, the min in (3.17) is minimized, for a given $C_0 = \sum_{i=1}^N \lambda_i$, when the two bounds coincide. This gives the fundamental lower bound

$$t_{M+1} \geq \frac{(M^2 C_0 - M^2 \lambda_{M+1} + 2M C_0 - M \lambda_{M+1} + C_0) |F|}{C_0 \left((M+1) \left(\sum_{i=0}^M C_i \right) + M \sum_{i=M+2}^N C_i \right)}. \quad (3.21)$$

When $C_0 > C_0^*$, the value of $\sum_{i=1}^M \lambda_i$ necessary to achieve this bound violates (3.13). In this case, the algorithm sets λ_i , $i < M$, to its upper bound of $C_i / (M-1)$, and (3.16) becomes $|\cup_{i=M+2}^N R_{i,M+1}(t_M, t_{M+1})| = \sum_{i=M+2}^N C_i t_{M+1}$.

When $C_0 > C_0^*$, nodes $i \in \{M + 2, \dots, N\}$ need not upload all of their information $F_i(t_M)$ to node M to achieve the lower bound (3.16); it is sufficient that λ_i , $i \in \{M + 2, \dots, N\}$, be large enough that $r_{i,M+2}(t) = C_i$ for all $t \in (t_M, t_{M+1})$. The LP (3.2) ensures that condition is met, while sequentially providing as much server capacity on $(0, t_M)$ as possible to nodes $M + 2, \dots, N$.

In either case, Algorithm 3.1 achieves the lower bound on t_{M+1} while maintaining $t_1, \dots, t_M = |F|/C_0$.

Finally, we claim after t_{M+1} , each node i receives at rate $C - C_i$ on its finishing interval, and C_{i-1} on the previous interval. To confirm, consider the fictional time interval when another node $k \notin \{1, \dots, N\}$, needs to receive all information held by nodes $\{1, \dots, N\}$ (i.e., it has no portion of the file). In this case, the amount of time it takes to transmit if all nodes have access to the entire file, $|F|/C$, is less than the amount of time it takes for any individual node to upload its assigned portion of the file, $\lambda_i t_1 / C_i$.

Under the algorithm,

$$\lambda_N \leq \lambda_i, \quad \forall i \in \{1, \dots, N\}, \quad (3.22)$$

including in the case that $C_0 > C_0^*$. To show that each node has enough information to transmit fully on any time interval, it is sufficient to show that

$$\frac{\sum_{i=1}^M \lambda_i}{C_0 + \sum_{i=1}^N C_i} \leq \frac{\lambda_N}{C_N} \quad (3.23)$$

which can be reformed as

$$\frac{C_0}{C} \leq \frac{\sum_{i=M+2}^N \lambda_i}{\sum_{i=M+2}^N C_i} \quad (3.24)$$

and results in a bound of

$$C_0 \geq \frac{C_{M+1}(C - C_0)}{MC_{M+1} + \sum_{i=M+2}^N C_i}. \quad (3.25)$$

This lower bound on C_0 for the condition to hold is strictly less than the lower bound due to the multiplicity constraint. Thus, full utilization is maintained for all time intervals prior to (t_{N-1}, t_N) when following the suggested optimal scheme. ■

The min-min proof will prove to be an important step in determining the scheme for minimizing the average finish time.

CHAPTER 4
AVERAGE TIME CONCLUSIONS

The min-min result, though clearly useful in achieving lower average finish times with minimal impact on the optimal last finish time, does not have a clear analytical meaning on its own. Its situational usefulness is not certain without connection to another, more analytically tractable metric. Luckily, a clear connection to average finish time is possible.

We now show that the min-min result, achieved by the scheduling algorithm in section 3.1, also minimizes the average finish time. This is consistent with the intuition of approximating “shortest job first” scheduling in a scenario complicated by the presence of multiple servers.

Theorem 2 *Min-min times minimize average finish time.*

Proof of Theorem 2 *Assume that for all $t \geq \underline{t}_{M+1}$, any node i can send any piece of information $R \in F$ to any other node j , regardless of if the condition $R \in F_i(t)$ is met, as long as $i \neq j$. All other rate and data identity constraints are assumed to hold.*

Also, while t_i is defined to be the time when node i finishes receiving the entire file, let a_k be the time at which k nodes have finished receiving the entire file.

The proof is separated into four parts.

Min-Min Implies Min-Average for Subset of Nodes

M = 1 or M > 1, $C_0 \leq C_0^*$

In the case where $M = 1$, or $M > 1$ and $C_0 \leq C_0^$, the finish times t_1, \dots, t_M are all minimized to bottleneck time by the min-min scheme, and similarly a_1, \dots, a_M are*

minimized. Additionally, (3.9) and (3.21), the absolute lower bounds on t_2 and t_{M+1} , respectively, are achieved. Inequalities (3.9) and (3.21) are based on expressions of the amount of information which can be sent into a set of $M + 1$ nodes by a time at which $M + 1$ nodes have finished. By ordering the capacities such that the $M + 1$ highest capacity nodes finish first, these expressions are maximized, and thus result in the lowest possible bounds on a_{M+1} . Since all bounds a_1, \dots, a_{M+1} are achieved, the minimum possible sum

$$\sum_{i=1}^{M+1} a_i$$

is achieved.

$M > 1, C_0 > C_0^*$

The maximum amount of information which can go into set $1, \dots, M + 1$ by time t_{M+1} is

$$\begin{aligned} & \left(C_0 + \sum_{i=1}^M C_i \right) t_{M+1} + C_{M+1} t_M - \sum_{i=M+2}^N F_i(t_{M+1}) \\ & + \min \left(t_{M+1} \sum_{i=M+2}^N C_i, (M + 1) \sum_{i=M+2}^N F_i(t_{M+1}) \right) \end{aligned} \quad (4.1)$$

As shown earlier, this expression is maximized for any t_M and t_{M+1} when

$$t_{M+1} \sum_{i=M+2}^N C_i = (M + 1) \sum_{i=M+2}^N F_i(t_{M+1}) \quad (4.2)$$

where $\sum_{i=M+2}^N F_i(t_{M+1})$ is considered a “design parameter”. Let

$$\sum_{i=M+2}^N F_i^*(t_{M+1})$$

be the value which solves (4.2).

Now, consider increasing $\sum_{i=M+2}^N F_i^*(t_{M+1})$ by X , resulting in a similarly-constructed bound on t_{M+1} of

$$t_{M+1} \geq \frac{(M + 1)|F| - C_{M+1} t_M + \sum_{i=M+2}^N F_i^*(t_{M+1}) + X}{C - C_{M+1}}. \quad (4.3)$$

Separating out the portion of $F_i^*(t_{M+1})$ corresponding to data received after t_M ,

$$t_{M+1} \geq \frac{(M+1)|F| - C_{M+1}t_M + \sum_{i=M+2}^N F_i^*(t_M) + (F_i^*(t_{M+1}) - F_i^*(t_M)) + X}{C - C_{M+1}}. \quad (4.4)$$

A bound on t_M constructed in similar fashion is

$$t_M \geq \frac{M|F| - C_M t_{M-1} - (M-1)(F_{M+1}(t_M) + \sum_{i=M+2}^N F_i^*(t_M) + X)}{C - C_M - \sum_{i=M+1}^N \frac{C_i}{M}}. \quad (4.5)$$

Since the sum $\sum_{i=1}^M t_i$ cannot be reduced below $M|F|/C_0$, t_{M-1} is fixed to be $|F|/C_0$. Summing the two bounds, a bound on $t_M + t_{M+1}$ is determined. The derivative of this lower bound with respect to X is

$$\frac{-(M-1)}{C - C_M - \sum_{i=M+1}^N \frac{C_i}{M}} + \frac{1 + \frac{-C_{M+1}(M-1)}{C - C_M - \sum_{i=M+1}^N \frac{C_i}{M}}}{C - C_{M+1}} \quad (4.6)$$

which can be simplified to

$$\frac{-(M-1)(C - C_{M+1}) + (C - C_M - \sum_{i=M+1}^N \frac{C_i}{M}) - (M-1)C_{M+1}}{(C - C_M - \sum_{i=M+1}^N \frac{C_i}{M})(C - C_{M+1})} \quad (4.7)$$

the numerator of which is clearly negative. So, for increasing values of X for which nodes $M+1, \dots, N$ do not receive more information than $\sum_{i=M+1}^N C_i |F| / (MC_0)$, the sum $\sum_{i=1}^{M+1} t_i = \sum_{i=1}^{M+1} a_i$ decreases. This implies that the min-min solution minimizes $\sum_{i=1}^{M+1} t_i = \sum_{i=1}^{M+1} a_i$.

Performance Bounds for Late-Finishing Nodes

Assume that a vector $\mathbf{F}(\underline{t}_{M+1})$ represents the amount of the file which is held by all nodes at time \underline{t}_{M+1} . Given $\mathbf{F}(\underline{t}_{M+1})$, as well as the relaxation of the data identity constraint in the initial assumption, a clear lower bound on the time of the next node to finish is

$$a_i \geq \min_{j \in B} \frac{|F| - |F_j(t_{M+1})|}{C - C_j} \quad (4.8)$$

where B is the set of nodes which have not finished at time \underline{t}_{M+1} .

Lower bounds on a_k , $k > i$, can also be shown inductively.

$$a_k \geq \min_{j \in B'} \frac{|F| - |F_j(t_{M+1})| - \hat{C}_{k-1}(a_{k-1} - a_{k-2})}{C - C_j} \quad (4.9)$$

where B' is the set of nodes which haven't finished by time a_{k-1} and \hat{C} is the capacity of the last node to finish.

By summing these a_k bounds, a bound on the sum of the late-finishing nodes can be derived. We will show that when $\mathbf{F}(\underline{t}_{M+1})$ is the result of the min-min scheme, this sum bound is minimized by the min-min order and achieved by the min-min scheme.

Order for Achieving Bounds on Late-Finishing Nodes

Assume that the vector $\mathbf{F}(\underline{t}_{M+1})$ is the result of following the min-min scheme in section 3 up to time \underline{t}_{M+1} . The bound in (4.8) is minimized when selecting the next node to finish based on minimizing the quantity

$$\frac{|F| - |F_j(t_{M+1})|}{C - C_j}.$$

Consider nodes K and L such that $M + 1 < K < L \leq N$. Excepting the contribution

of node $M + 1$ on the time interval $(\underline{t}_M, \underline{t}_{M+1})$, which can be directed to either node, we seek to show that the potential a_i bounds can be compared as follows:

$$\frac{|F| - \frac{|F|}{C_0} \lambda_K}{C - C_K} \leq \frac{|F| - \frac{|F|}{C_0} \frac{C_L}{C_K} \lambda_K}{C - C_L} \quad (4.10)$$

where the substitution for λ_L is justified by (3.3) and (3.4). (The added complexity from the consideration of the LP (3.2) is not considered here, as it only amplifies the benefit of the min-min approach. Not including it is a worst-case analysis.)

This comparison can also be written as

$$\frac{C_0 - C_K \Phi}{C - C_K} \leq \frac{C_0 - C_L \Phi}{C - C_L}$$

where $\Phi \in (\frac{1}{M+1}, \frac{1}{M}]$, based on the necessary condition for achieving minimum t_M .

Rewriting this expression as

$$\frac{\frac{C_0}{\Phi} - C_K}{C - C_K} \leq \frac{\frac{C_0}{\Phi} - C_L}{C - C_L}$$

shows that the inequality will hold as long as the condition

$$\frac{C_0}{\Phi} \leq C \quad (4.11)$$

is met. Lemma 1 only guarantees that the inequality holds for a portion of the possible range of Φ ; additional justification is needed.

For $M > 1$, when $\Phi = 1/M$, (4.11) can be re-written as

$$C_0 \leq \frac{\sum_{i=1}^N C_i}{M - 1}$$

which is clearly included in the range covered by Lemma 1. Consider the same bound as Φ decreases toward $1/(M + 1)$.

$$C_0 \leq \frac{\sum_{i=1}^N C_i}{M}$$

In between, the necessary upper bound on C_0 is

$$C_0 \leq \frac{\Phi \sum_{i=1}^N C_i}{1 - \Phi}, \quad (4.12)$$

the derivative of which is lower-bounded by $\sum_{i=1}^N C_i$. From (3.4), the derivative of C_0 with respect to Φ is at most $\sum_{i=1}^N C_i$. Since the upper bound on C_0 by min-min scheduling is below (4.11) as a result, the ordering condition for min-min holds for finishing times a_{M+2}, \dots, a_N .

Since the $M > 1$ condition is an upper bound on Φ values for the $M = 1$ condition, min-min ordering is optimal for all multiplicities given that the min-min scheme is used up to time \underline{t}_{M+1} .

Necessity of Minimizing Ordered-Subset Average Time

We now show that minimizing $\sum_{i=1}^{M+1} t_i$ is necessary for minimizing $\sum_{i=1}^N t_i$.

First consider the possibility of data given to nodes $1, \dots, M + 1$ on $(0, \underline{t}_{M+1})$ in the min-min strategy instead being given to nodes $M + 1, \dots, N$ on that same interval. For data of size ϵ , where ϵ is not necessarily small, this will result in an increase to the lower bound of $\sum_{i=1}^{M+1} t_i$ of at least $\epsilon/(C - C_{M+1})$ and a decrease to $\sum_{i=M+2}^N t_i$ of at most $\epsilon/(C - C_{M+2})$. Similarly, there is no benefit to shifting data from the set $M + 2, \dots, N$ to the set $1, \dots, M + 1$, as the minimum possible value of $\sum_{i=1}^{M+1} t_i$ has already been achieved.

Next, note that in order for nodes $M + 2, \dots, N$ to share

$$t_{M+1} \sum_{i=M+2}^N C_i$$

data with nodes $1, \dots, M + 1$ on $(0, t_{M+1})$ while only holding

$$\sum_{i=M+2}^N \frac{C_i}{M + 1} t_{M+1}$$

data, each node j must transmit at rate C_j for the entire interval $(0, t_{M+1})$. So, there is only one allocation of data among nodes $M + 2, \dots, N$ that allows for achieving the lower bound on $\sum_{i=1}^{M+1} a_i$ when $C_0 \leq C_0^*$. As discussed in the previous section, more flexibility exists when $C_0 > C_0^*$, with this flexibility absorbed into the LP (3.2).

Finally, consider an ϵ shift of data between two nodes in the set $M + 2, \dots, N$ prior to time t_{M+1} . In the best case, this will result in an increase to the lower bound of $\sum_{i=1}^{M+1} t_i$ of at least

$$\frac{\epsilon}{C_0 + C_{M+2} + \sum_{i=1}^M C_i} \quad (4.13)$$

since only nodes $0, 1, \dots, M$ and a single node in the set $\{M + 2, \dots, N\}$ will have the necessary data for completing node $M + 1$. The decrease in $\sum_{i=M+2}^N t_i$ is at most

$$\frac{\epsilon}{C - C_{M+2}} - \frac{\epsilon}{C - C_N} \quad (4.14)$$

where the first term comes from the addition of information to the node which is serviced at the slowest rate, and the negative term comes from removal of information from the node which is serviced at the fastest rate.

Since (4.13) is larger than (4.14), no decrease in the sum result $\sum_{i=1}^N t_i$ can be achieved by shifts prior to time t_{M+1} . Since at that point, any node j can be serviced at rate $C - C_j$, sequential minimization is optimal in the average sense. ■

4.1 Advantage of Heterogeneity

Now we demonstrate an application of our results by studying how the heterogeneity of peer capacities affect the minimal average finish time. The algorithm in section 3.1 (now proved to be optimal) is used to calculate the minimal average finish time.

Example 3: *Heterogeneity improves performance.*

The same network in Example 1 is used. The sum of peer capacities is fixed ($\sum_{i=1}^4 C_i = 50$) the server capacity is 100 and the file size is 10000. According to (2.1), the last finish time does not change since the sum capacity is fixed. However, the average finish time changes when the heterogeneity of peer capacities change. In Fig. 4.1, the average finish time is plotted against the variance of the peer capacities.

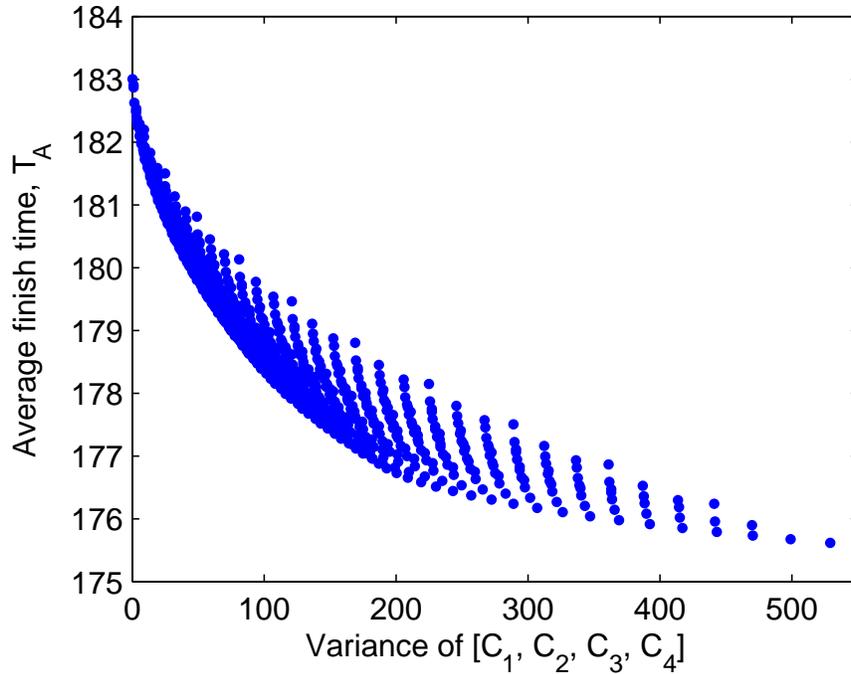


Figure 4.1: The optimal average finish time for $N = 4$, $C_0 = 100$, $\sum_{i=1}^4 C_i = 50$, and $|F| = 10000$, with C_i values restricted to integers > 1 [5]. (©2009 IEEE)

Interestingly, we see that increasing heterogeneity (larger variance) in general decreases the average finish time (better performance). The reason is that the capacity available to send a particular fragment of the file grows quickest when sent to a node with large capacity, as opposed to being split and sent to multiple nodes with smaller capacities. This effect outweighs the diminished rate at which lower capacity nodes can send received information.

CHAPTER 5

A LOOK AT THE DOWNLOAD-CONSTRAINED CASE

Thus far, we have considered only the situation where the rate at which nodes can send information is constrained and the rate at which they can receive information is constrained. While the scenario in which both upload and download constraints are in place is too complex for analysis at this time, analysis of a solely download-constrained scenario sheds some light on both the complexity of the general case and its possible solutions.

Consider the same problem as in the upload-constrained case, except now the constraints C_1, \dots, C_N apply to receive rates only. Assume that the server capacity C_0 remains an upload constraint, and that $\sum_{i=1}^N C_i > C_0 > C_N$. (Outside of this range, the problem is trivial.) We consider only the client-server model, ignoring any upload capability at nodes $1, \dots, N$.

In the $N = 2$ case, this problem can be represented similarly to the upload-constrained case.

$$\begin{aligned} \min \quad & t_1 + t_2 \\ \text{where if } \lambda > 0.5 \quad & t_1 = |F| / \min \lambda C_0, C_1 \\ & t_2 = (|F| - \min(1 - \lambda) * C_0, C_2 * t_1) / \min C_0, C_2 + t_1 \\ \text{else} \quad & \\ & t_1 = (|F| - \min(\lambda) * C_0, C_1 * t_2) / \min C_0, C_1 + t_2 \\ & t_2 = |F| / \min(1 - \lambda) * C_0, C_2 \end{aligned}$$

Here, a fixed resource is available to all nodes, but each node is not necessarily able to access that resource in full. Any strategy which results in all C_0 capacity being used for all time will minimize the last finish time T_L . However, it is not immediately clear what strategy will minimize the average finish time T_A .

Depending on the download constraints, particularly that of node N , which is defined to have the smallest download rate constraint, it is possible that more than one strategy will minimize the average finish time. The following lemma illustrates a strategy which reliably minimizes the average finish time.

Lemma 2 *If finish times are minimized in sequentially increasing order of capacity, with unutilized capacity from the minimization of any node i devoted to node $i - 1$, then the optimal average finish time T_A^* is achieved.*

Proof of Lemma 2 *Let t_i be the time at which any i nodes have finished receiving the entire file. Any t_i is bounded below by*

$$\frac{|F|i}{C_0}.$$

This results in a lower bound on $\sum t_i$ of

$$\sum_{i=1}^N \frac{|F|i}{C_0} = \frac{N(N+1)}{2} \frac{|F|}{C_0}.$$

However, given a_i , the time when node i begins receiving information from the server, the download constraint implies the bound

$$b_i - a_i \geq \frac{|F|}{C_i}$$

where b_i is the finish time of the node with the i th largest download rate. This additional constraint results in the bound

$$\sum_{i=1}^N t_i \geq \frac{N(N+1)}{2} \frac{|F|}{C_0} + \sum_{i=1}^N \frac{|F|}{\min C_0, C_i} - \frac{|F|}{C_0}$$

This bound is achieved by the sequential minimization strategy. ■

At first glance, the result of Lemma 2 would seem to indicate that when download constraints are introduced, the optimal solutions of the upload and download-constrained cases would conflict; the upload constraints imply sequential minimization with decreasing capacity, and the download constraints imply sequential minimization with increasing capacity. However, it should be noted that the solution provided by Lemma 2 is not necessarily the only solution for the download constrained case; it reliably solves the problem, but other variants may also solve the problem.

For example, in a system with large N , the ordering of the first few nodes is actually irrelevant in trying to minimize the average finish time. Order is only important when considering the last nodes, as unutilized capacity due to download constraints when finishing the last node is crucial to minimizing the average finish time. As such, choosing to follow a modification of the min-min strategy which adjusts the initial server allocation vector λ to consider download constraints but effectively proceeds according to the sequential minimization strategy in descending order of capacities should achieve average finish time arbitrarily close to the optimal average finish time.

CHAPTER 6

CONCLUSION

This paper has considered the transmission scheduling issue in an upload-constrained peer-to-peer file distribution system. Under the assumptions that the network is static and that the file is infinitely divisible, an explicit transmission scheduling algorithm has been proposed which provably minimizes the average finish time for all peers. New inductive concepts like min-min times and novel techniques such as water-filling are used in obtaining the result. Some basic results for the download-constrained case were considered, and conclusions were drawn for how the upload- and download-constrained cases might interact.

Several of the assumptions put in place to make this problem tractable can be removed with minimal effect on the result. For example, many similar problems in the literature are evaluated based on the principle that the file of interest is broken into small pieces, but it is not necessarily assumed that these pieces can be made infinitesimally small. In trying to minimize the average finish time, the same approach as described herein would be taken, though the discrete nature of the file would have to be considered. In this situation, slightly less optimal results would be achieved. Another example of constraint modification which would have minimal effect would be the inclusion of a download constraint for cases where N is small.

There are a number of related directions in which to extend this work. First, it would be useful to investigate fully how the optimal results change when download constraints are introduced. An ultimate goal for analyzing this sort of problem would be an understanding of the tradeoff in the average finish time between upload and download constraints if *total capacity* at a node is a constrained quantity with a flexible upload/download ratio. Second, understanding the behavior of our optimal scheduling

when nodes dynamically enter and leave upon completion [8, 28] would be necessary before its application in practice. Similarly, stochastic considerations relating to the upload or download capacity could modify the optimal approach.

Another interesting direction is to look at similar optimality results under peer-to-peer streaming [7, 9] context. Finally, this paper only gives the best possible centralized solution without any coding. Exploring corresponding distributed solutions or the effect of tools like network coding [10, 3, 14] can be potentially fruitful.

BIBLIOGRAPHY

- [1] J. M. Almeida, D. L. Eager, M. Ferris, and M. K. Vernon. Provisioning content distribution networks for streaming media. *Proceedings of IEEE Infocom*, 2002.
- [2] J. Chan, V. Li and K. Lui. Performance comparison of scheduling algorithms for peer-to-peer collaborative file distribution. *IEEE Journal on Selected Areas in Communications*, 25(1):146–154, January 2007.
- [3] P. Chou and Y. Wu. Network coding for the Internet and wireless networks. *IEEE Signal Processing Magazine*, 24(5):77–85, September 2007.
- [4] G. M. Ezovski, L. L. H. Andrew, A. Anandkumar, and A. Tang. Minimizing Average Finish Time in P2P Networks. to appear in *Proceedings of Allerton Conference on Comm., Control, and Comp.* 2008.
- [5] G. M. Ezovski, A. Tang and L. L. H. Andrew. Minimizing Average Finish Time in P2P Networks. to appear in *Proceedings of IEEE Infocom 2009*. Available <http://www.people.cornell.edu/pages/gme8/icomsubmit09.pdf>.
- [6] B. Fan, D. Chiu and J. Lui. The delicate tradeoffs in Bit Torrent-like file sharing protocol design. In *Proceedings of IEEE ICNP*, 2006.
- [7] L. Gao, D. Towsley and J. Kurose. Efficient schemes for broadcasting popular videos. In *Proceedings of ACM NOSSDAV*, 1998.
- [8] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of ACM SIGCOMM*, 2003.
- [9] X. Hei, C. Liang, Y. Liu and K. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):1672–1687, December 2007.
- [10] T. Ho, M. Médard and R. Koetter. An information-theoretic view of network management. *IEEE Transactions on Information Theory*, 51(4):1295–1312, April 2005.
- [11] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, M. Faloutsos. “File-sharing in the Internet: A characterization of P2P traffic in the backbone.” Technical Report, UC Riverside, 2003.

- [12] R. Kumar and K. Ross. Peer-assisted file distribution: The minimal distribution time. In *Proceedings of IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2006.
- [13] J. Kurose and K. Ross. *Computer Networking*. Fourth edition, Addison Wesley, 2007.
- [14] S. Li, R. Yeung and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003.
- [15] M. Lingjun and K. Liu. Scheduling in P2P file distribution – On reducing the average distribution time. In *Proceedings of Consumer Communications and Networking Conference*, 2008.
- [16] S. Liu, R. Shen, W. Jiang, J. Rexford and M. Chiang. Performance bounds for peer-assisted live streaming. *Proceedings of ACM SIGMETRICS*, 2008.
- [17] L. Massoulié and M. Vojnović. Coupon Replication Systems. *IEEE/ACM Transactions on Networking*, 16(3):603–616, June 2008.
- [18] M. Mehyar. Distributed Averaging and Efficient File Sharing on Peer-to-Peer Networks. Doctoral Thesis, California Institute of Technology, 2006.
- [19] M. Mehyar, G. WeiHsin, S. Low, M. Effros and T. Ho. Optimal strategies for efficient peer-to-Peer file sharing. *Proceedings of IEEE ICASSP*, 2007.
- [20] J. Munding, R. Weber and G. Weiss. Analysis of peer-to-peer file dissemination amongst users of different upload capacities. *ACM SIGMETRICS Performance Evaluation Review*, 34(2):5-6, September 2006.
- [21] J. Munding, R. Weber and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. *Journal of Scheduling*, 11(2):1094-6136, April 2008.
- [22] Q. Ou and D. Tsang. An optimal bandwidth allocation algorithm for file distribution network. In *Proceedings of ChinaCom*, 2007.
- [23] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips. The Bittorrent P2P file-sharing system: Measurements and analysis. *Proceedings of 4th International Workshop on Peer-to-Peer Systems*, 2005.
- [24] D. Qiu, and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *Proceedings of ACM Sigcomm*, 2004.

- [25] S. Sanghavi, B. Hajek and L. Massoulié. Gossiping with multiple messages. *IEEE Transactions on Information Theory*, 53(12):4640–4654, December 2007
- [26] I. Stoica, R. Morris, D. Karger, M. Kaashoek and H. Balakrishnan. A scalable peer-to-peer lookup service for Internet applications. *Proceedings of ACM SIGCOMM*, 2001.
- [27] X. Yang and G. De Veciana. Service capacity of peer to peer networks *Proceedings of IEEE Infocom*, 2004.
- [28] Z. Yao, D. Leonard, X. Wang and D. Loguinov. Modeling heterogeneous user churn and local resilience of unstructured P2P networks. In *Proceedings of IEEE ICNP*, 2006.
- [29] X. Zheng, C. Cho and Y. Xia. Optimal peer-to-peer techniques for massive content distribution. In *Proceedings of IEEE Infocom*, 2008.