This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

Minimizing Average Finish Time in P2P Networks

G. Matthew Ezovski and Ao Tang
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853

Lachlan L.H. Andrew
Centre for Advanced Internet Architecture
Swinburne University of Technology
Hawthorn, Vic 3122, Australia

***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 paper, 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 argued that this scheduling also potentially minimizes average finish time for the network. This 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.**

I. Introduction

Peer-to-peer (P2P) networking utilities are 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 [19], [20] have shown that upwards of 45% of Internet traffic can be attributed to P2P applications. They have also generated a great deal of research activity in the last couple of years; see e.g., [3]–[6], [8], [12], [20], [22], [24] 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 [9], 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 finding an explicit scheduling procedure to achieve optimal average finish time in a fully-connected P2P network.

Several papers have explored performance of P2P networks [1], [8], [13], [15], [17], [21], [23], [25]. Ones that deal with various optimal scheduling algorithms are particularly related. For example, Mundinger et al. [16], [17] 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 [8] where optimal strategies is discussed for file distribution when multiple classes of service exist. Recently, Mehyar et al. [15] extend 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 small set of higher order cases. Building upon all these 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*. In other words, a whole file is needed as opposed to just that amounts of data, 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 [18] but results in 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 focuses on the problem of designing explicit file dissemination scheduling algorithms which minimize average finish time. To overcome the above mentioned difficulty, our overall strategy is to use an intermediate step by introducing another concept (min-min time), which has an inherent inductive structure that facilitates algorithm design. The paper is organized as follows. Section II reviews the solution that achieves optimal last finish time and then formulates the min-min and average finish time problems. After that, we present two main results. The first one is in section III, where an explicit solution to achieve the optimal min-min times is provided, along with a water-filling interpretation. The second one is in section IV, where we argue that achieving min-min times can potentially minimize the average finish time. We conclude in section V and discuss some possible extensions.

II. Problem Formulation

A. 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 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 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 $C_1 \geq C_2 \geq \ldots \geq C_N$.
- $C = C_0 + \sum_{i=1}^{N} C_i$: 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 $|F_i(t)| = |F|$
- $|F|/C_0$ – 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,

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

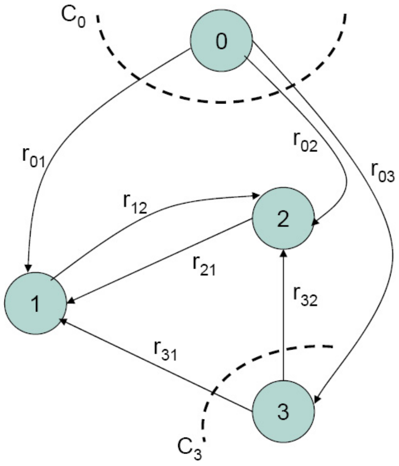

Fig. 1. A diagram showing the constraints on communication between nodes in a 3-node plus server configuration. The dashed lines represent the sum rate constraints $\sum_{j=0}^{N} r_{ij}(t) \leq C_i \ \forall i$.

The "data identity" constraint can now be expressed as

- $R_{ij}(t, t+\tau) \subseteq F_i(t+\tau)$ (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
- $\frac{d}{dt}|F_j(t)| = \sum_{i=0}^{N} r_{ij}(t) \quad \forall j, t.$

B. 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,

$$T_L^* \geq \max(|F|/C_0, N|F|/C) \tag{1}$$

Mundinger et al. [16] show that this lower bound is tight by looking at the following two possibilities.

1) 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 $(0, T_L^*)$.

2) 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.

It turns out that forcing all the nodes to finish receiving the file at T_L^* might artificially limit the performance of the network by other metrics. In other words, by 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. (It will be clear how to calculate this in later sections.) The results are summarized in Fig. 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

$$T_A = \frac{\sum_{i=1}^{N} t_i}{N}. \tag{2}$$

In general, to minimize the average finish time, we want 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 to even write down the optimization problem for general case. The following example illustrates this difficulty with a very simple 2-peer network.

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

Consider 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.

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

Here the data identity constraint forces us to keep track of the sizes of many distinct pieces of data even when $N = 2$ (the later 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 allows us not to optimize all data pieces at the same time. The min-min times that will be introduced in section II-C serve this role.

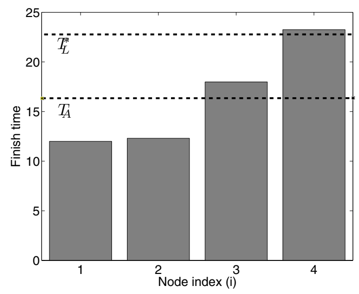

Fig. 2. Results for the $N = 4$ case, with $C_0 = 12$, $C_1 = 6$, $C_2 = 4$, $C_3 = 2$, $C_4 = 1$, and $|F| = 144$. T_A is the associated average finish time, and T_L^* is the optimal last finish time.

C. Min-Min Times

The min-min time sequentially minimizes the individual finish times. Besides its relation to the optimal average finish time, it is also of independent interest, since minimizing the completion times of early flows improves the robustness to disconnection of the network [11].

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 $i+$ 1st finish time, given that all previous finish times are minimized.

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 section III. Before delving into our main results, we introduce the useful concept of *multiplicity* [15], 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 $|F|/C_0$. The following lemma is proved in [14].

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

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

Then M is the largest integer such that

$$C_0 \le \sum_{i=1}^{M} \frac{C_i}{M-1} + \sum_{M+1}^{N} \frac{C_i}{M}. \tag{3}$$

III. Scheduling to Achieve Min-Min Times

When the multiplicity $M = N$, all nodes can finish by $|F|/C_0$ using the schedule reviewed in section II-B. We now study optimal schedules for the remaining cases ($M < N$).

The main difficulty in achieving min-min times is when we try to minimize t_i, how to use the extra capacities of some peers. It will be shown that they only need to minimize t_{i+1}. In other words, scheduling for more than one step ahead is not useful. Another difficulty is how to schedule all peers to minimize t_i given they all have different capacities C_i. 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.

More precisely, see Fig. 3(a) for an illustration. During the interval (t_{i-1}, t_i), the jth column has width C_j, and area

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

$F_j(t_1) \setminus F_i(t_{i-1})$. 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). The optimal way is to let nodes $0 \le j < i$ send the shaded data in Fig. 3(b), equalizing the finish times $t_i - t_{i-1} = |F_j(t_1) \setminus F_i(t_{i-1})|/C_j$.

The only remaining question is what should node i do when others are uploading data to it. Due to the "data identity" constraint, $r_{ii}(t) = 0$, and therefore it can't transfer data to itself. The optimal way is to use node i's capacity to send data to $i+1$. The exact data to sent will be determined by "helium-filling". for the following time interval, (t_i, t_{i+1}), in that data U_{ij} would have been in column j 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(b)). In later proofs, we will provide specialized water-filling figures for different cases (Figures 4 and 6).

The actual scheduling algorithm is stated in Algorithm 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

$$\frac{M(C_0^* - \frac{C_{M+1}}{M} - \sum_{i=1}^{M} \frac{C_i}{M-1}) + C_0^*}{C_0^* + \sum_{i=1}^{M} C_i} \tag{4}$$
$$= \frac{(M+1)\left(C_0^* - \frac{C_{M+1}}{M} - \sum_{i=1}^{M} \frac{C_i}{M-1}\right)}{\sum_{i=M+2}^{N} C_i}.$$

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 Algorithm 1 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.

$$\max \sum_{i=M+2}^{N} (N-i)\lambda_i \tag{5}$$

s.t.

$$\frac{C_i}{M+1} < \lambda_i \le \frac{C_i}{M} \quad \forall i \ge M+2$$
$$\sum_{i=M+2}^{N} \lambda_i = C_0 - \frac{C_{M+1}}{M} - \sum_{i=1}^{M} \frac{C_i}{M-1}$$
$$\frac{(M+1)\lambda_i - C_i}{C_i} \ge$$
$$\frac{\frac{1}{M-1}\sum_{i=1}^{M} C_i + (M+1)\sum_{i=M+2}^{N} \lambda_i - \sum_{i=M+2}^{N} C_i}{C - C_{M+1}}$$

The following theorem characterizes Algorithm 1.

Algorithm 1 Optimal scheduling to achieve min-min times

```
If M=1
  • On $(0, t_1)$, let $r_{0i} = \lambda_i$, $r_{i1} = \min(\lambda_i, c_i)$, $r_{i2} = c_i - \min(\lambda_i, c_i)$, where $\lambda_i$ satisfy
      $\sum_{i=1}^{N} \lambda_i = C_0 \qquad \lambda_2 = C_2 \qquad \frac{2\lambda_i}{C_i} = \frac{\lambda_1 + C_0}{C_0 + C_1}, \forall i > 2$   (6)
  • On $(t_{i-1}, t_i)$, $2 \le i < N$:
    $r_{ji}(t) = C_j \quad \forall j \ne i$, with $R_{ji}(t_{i-1}, t_i) \cap R_{ki}(t_{i-1}, t_i) = \emptyset$ and $(R_{ji}(t_{i-1}, t_i) \cup R_{ki}(t_{i-1}, t_i)) \cap F_i(t_{i-1}) = \emptyset$ for all $k \ne j$. Node $i$ sends data $U_{ij}$ to node $i+1$ with $r_{i,i+1}(t) = C_i$ such that
    1) $U_{ij} \cap U_{ik} = \emptyset$ for all $k \ne j$ (data is disjoint)
    2) $U_{ij} \in F_j(t_{i-1})$ (data is held at $t_{i-1}$ by node $j$)
    3) for all $j \ge 0$, $j \ne i+1$,
         $\frac{|U_{ij}|}{t_i - t_{i-1}} = \gamma_{ij} = \frac{C_j}{\left(\sum_{k=0, k \ne i+1}^{N} C_k\right)} C_i.$
Else
  • If $M = N-1$ or $C_0 \le C_0^*$, given by (4),
    Then let $\lambda_i$ solve
         $\sum_{i=1}^{N} \lambda_i = C_0$
         $\frac{\lambda_i}{C_i} = \begin{cases} \lambda_1/C_1 & \text{if } i \le M \\ 1/M & \text{if } i = M+1 \\ \lambda_{M+2}/C_{M+2} & \text{if } i \ge M+2 \end{cases}$
         $\frac{\lambda_{M+2}}{C_{M+2}} = \frac{M + \sum_{i=1}^{M} \lambda_i + C_0}{(M+1)\sum_{i=0}^{M} C_i}$
    Else let $\lambda_i = C_i/(M-1)$, $\forall i \le M$, and let $\lambda_i$ for $i \ge M+2$ satisfy the LP (5).
    EndIf
  • On $(0, t_M)$:
    $r_{0i} = \lambda_i, \forall i$; $r_{ij}(t) = \lambda_i$ for $j \le M, j \ne i$; $r_{ij}(t) = 0$ for $j > M+1$; $r_{i,M+1}(t) = C_i - \sum_{j \ne M+1} r_{ij}(t)$.
  • On $(t_{i-1}, t_i)$ for $M+1 \le i < N$:
    $r_{ji}(t) = C_j$ for $j \ne i$, such that
         $|R_{0i}(t_{i-1}, t_i) \cap F_j(t_{i-1})| = \mu_{ji}(t_i - t_{i-1}),$
    for $j < i$, where
         $\mu_{j,M+1} = \frac{C_0}{(\sum_{k=1}^{N} C_k) - C_{M+1}} C_j, \quad \forall j \ne M+1,$
         $\mu_{j,i} = \lambda_i t_1 - C_i \frac{(C_0 - \lambda_i)t_1}{C_0 + C - C_i} \quad \forall j \ne i \ne M+1.$
    Also $r_{i,i+1}(t) = C_i$, such that $|R_{i,i+1}(t_{i-1}, t_i) \cap F_j(t_{i-1})| = \gamma_{ji}$ where
         $\gamma_{ji} = \frac{C_i C_j}{(\sum_{k=1}^{N} C_k) - C_{i+1}}, \quad \forall j \ne i+1$
EndIf
On $(t_{N-1}, t_N)$, $r_{iN}(t) = C_i$ for $i < N$, and $r_{ik}(t) = 0 \quad \forall k$.
```

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

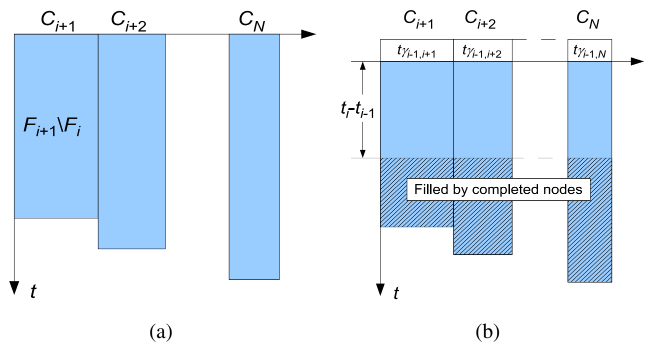

Fig. 3. Water filling. The width of column j is capacity C_j, and the depth is the time to transmit $F_j(t_1)\backslash F_i(t_{i-1})$ at rate C_j. In (a), node N takes longer to transmit its information. In (b), the server has *water-filled*, decreasing the time for all to complete transmission to node 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.

Theorem 1. *Algorithm 1 achieves min-min times.*

Proof: We state the proof for $M = 1$ here; the proof for $1 < M < N$ can be found in Appendix A.

Note first that Algorithm 1 is feasible. In particular, until time t_i, all nodes $j > i$ have disjoint data, while nodes $j < i$ have all data. Similarly, $U_{i,j}$ 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 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 $3, \ldots, N$. As in [15], the amount of information that can go into nodes 1 and 2 on $(0, t_2)$ is bounded above as

$$F_1(t_2) + F_2(t_2) \leq (C_0 + C_1)t_2 + C_2 t_1 + \frac{C'}{2} t_2. \tag{7}$$

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

$$\min\{C_i t_2, F_i(t_1) + F_i(t_2)\} - F_i(t_2). \tag{8}$$

Note that

$$\min\{C_i t_2, F_i(t_1) + F_i(t_2)\} \leq \frac{C_i t_2 + 2F_i(t_2)}{2} \tag{9}$$

with equality only if

$$2F_i(t_1) = 2F_i(t_2) = C_i t_2. \tag{10}$$

Substituting (9) into (8) and summing over $i \geq 3$ gives $C'T_2/2$, establishing (7).

A lower bound on $\underline{t}_2$ results from substituting $F_1(t_2) + F_2(t_2) = 2F$ into (7), and substituting the known value $\underline{t}_1 = |F|/C_0$, giving

$$\underline{t}_2 \geq \frac{2|F| - C_2|F|/C_0}{C_0 + C_1 + C'/2}. \tag{11}$$

This is achieved by Algorithm 1.

To see that the choice of λ_i is the only one which achieves $\underline{t}_2$, note that (10) 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

$$\frac{2\lambda_i}{C_i} = \frac{\underline{t}_2}{\underline{t}_1} \tag{12}$$

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

$$\frac{\lambda_1 + C_0}{C_0 + C_1} = \frac{\underline{t}_2}{\underline{t}_1}. \tag{13}$$

Combining (12) and (13) shows that λ_i, $i > 2$, must satisfy (6) to achieve $\underline{t}_2$. Thus, Algorithm 1 achieves $\underline{t}_1$ and $\underline{t}_2$, and (6) are necessary for any scheme which does.

Given that (6) must hold in order to achieve $\underline{t}_1$ and $\underline{t}_2$, 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) $\underline{t}_i$ is tightly bounded below by

$$\underline{t}_i \geq \frac{|F| - \lambda_i \underline{t}_1 - C_{i-1}(\underline{t}_{i-1} - \underline{t}_{i-2})}{C - C_i} + \underline{t}_{i-1}. \tag{14}$$

The term $\lambda_i \underline{t}_1$ is the amount of data received by node i from the server during the first interval, $(0, \underline{t}_1)$, and the term $C_{i-1}(\underline{t}_{i-1} - \underline{t}_{i-2})$ is the data received from node $i - 1$ in

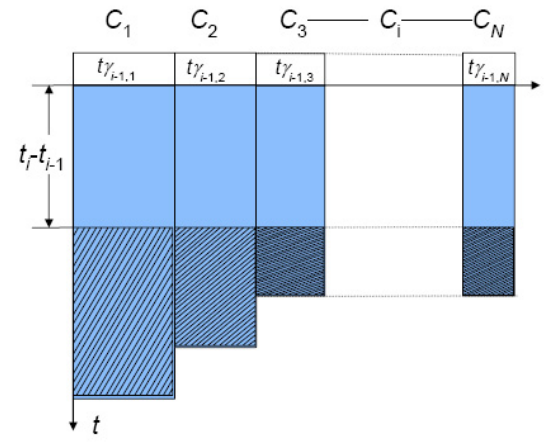

Fig. 4. A visual depiction of the waterfilling argument for the case when $M = 1$. Note that the bottoms of columns $M + 2, \ldots, N$ are level.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

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}). Algorithm 1 satisfies that and hence establishes the inductive step. ∎

IV. Achieving min-min times implies minimizing average finish time

We now argue that the min-min result, achieved by Algorithm 1 in section III, also potentially minimizes the average finish time. This is consistent with the intuition of approximating "shortest job first" scheduling, but is complicated by the presence of multiple servers.

Claim 1. *Min-min times minimize average finish time.*

Assume $C_0 \leq {C_0}^*$ (the argument for the case with $C_0 > {C_0}^*$ is in Appendix B). Let set A be nodes $1, \ldots, M, M+1$. The maximum amount of information which can go into set A by time t_{M+1} can be written as

$$\left(\sum_{i=0}^{M} C_i\right) t_{M+1} + C_{M+1} t_M - \sum_{i=M+2}^{N} F_i(t_{M+1}) \qquad (15)$$
$$+ \min\left(t_{M+1} \sum_{i=M+2}^{N} C_i, (M+1) \sum_{i=M+2}^{N} F_i(t_{M+1})\right)$$

This expression is maximized for any t_M and t_{M+1} when

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

with $\sum_{i=M+2}^{N} F_i(t_{M+1})$ as a parameter.

To achieve t_{M+1}, (15) must be larger than $(M+1)|F|$. Setting the inequality and solving for t_{M+1}, we find a bound on t_{M+1} of

$$t_{M+1} \geq \frac{(M+1)|F| - C_{M+1} t_M}{C - C_{M+1} - \sum_{i=M+2}^{N} \frac{C_i}{M+1}} \qquad (16)$$

Note also that from the multiplicity theorem, an achievable lower bound on $\sum_{i=1}^{M} t_i$ is $M|F|/C_0$. This is achieved by Algorithm 1 with the particular multiplicity M.

We now argue 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 $1, \ldots, M+1$ on $(0, t_{M+1})$ being instead given to nodes $M+1, \ldots, N$ on that same interval. For data of size ϵ, this will result in an increase to $\sum_{i=1}^{M+1} t_i$ of at least $\epsilon/(C - C_{M+1})$ and a decrease to $\sum_{i=M+2}^{N} t_i$ of at most $\epsilon/(C - C_{M+2})$ ignoring any cascading effects on future times. (The resulting structure from minimizing $\sum_{i=1}^{M+1} t_i$ allows for any remaining node to be serviced at full rate at any time ... thus producing these adjustment figures.)

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

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

data with nodes $1, \ldots, M+1$ 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 $M+2, \ldots, N$ that allows for achieving the lower bound on $\sum_{i=1}^{M+1} t_i$.

Finally, consider whether changes in these allocations could decrease $\sum_{i=M+2}^{N} t_i$ more than they increase $\sum_{i=1}^{M+1} t_i$. Consider again an ϵ shift of data between two nodes in the set $M+2, \ldots, N$ prior to time t_{M+1}. In the best case, this will result in an increase in $\sum_{i=1}^{M+1} t_i$ of at least

$$\frac{\epsilon}{C_0 + C_{M+2} + \sum_{i=1}^{M} C_i} \qquad (17)$$

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

$$\frac{\epsilon}{C - C_{M+2}} - \frac{\epsilon}{C - C_N} \qquad (18)$$

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 (17) is larger than (18), 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.

Before conclusion, we demonstrate an application of our results by studying how the heterogeneity of peer capacities affect the minimal average finish time. Algorithm 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 (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. 5, the average finish time is plotted against the variance of the peer capacities. 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.

V. 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 can potentially minimize 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.

There are a number of related directions in which to extend this work. First, it would be useful to investigate how the optimal results change when download constraints are introduced. Second, understanding the behavior of our optimal scheduling when nodes dynamically enter and leave upon completion [5], [24] would be necessary before its application in practice. Another interesting direction is to look at similar optimality results under peer-to-peer streaming [4], [6] context. Finally, this paper only gives a centralized solution without any coding. Exploring corresponding distributed solutions or the effect of tools like network coding [2], [7], [10] can be potentially fruitful.

VI. Acknowledgements

A subset of this work was presented at the Allerton 2008 workshop. This work was performed while L. Andrew was at Caltech.

Appendix A
Proof of Theorem 1 with $1 < M < N$

Proof: 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

$$\lambda_i \le \frac{C_i}{M-1} \qquad \forall i \le M \tag{19}$$

$$\lambda_i \le \frac{C_i}{M} \qquad \forall i > M \tag{20}$$

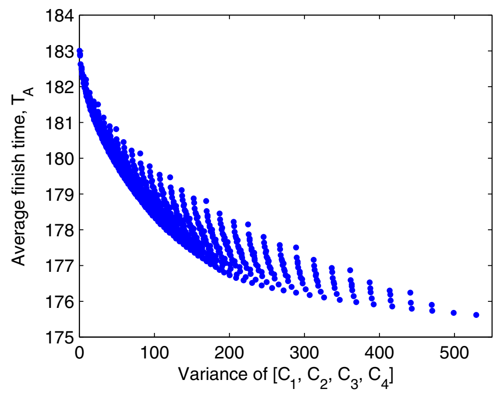

Fig. 5. the optimal average finish time for $N = 4$, $C_0 = 100$, $\sum_{i=1}^{4} C_i = 50$, and $|F| = 10000$, with C_i values restricted to integers > 1.

Since Algorithm 1 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 $\{M+2, \dots, N\}$:

$$|R_{0,M+1}(0, t_M)| = \lambda_{M+1} t_M$$

$$\left| \bigcup_{i=1}^{M} R_{i,M+1}(0, t_M) \right| \le \left(\sum_{i=1}^{M} C_i - (M-1) \sum_{i=1}^{M} \lambda_i \right) t_M$$

$$|R_{M+1,M+1}(0, t_M)| = 0$$

$$\left| \bigcup_{i=M+2}^{N} R_{i,M+1}(0, t_M) \right| \le \left(\sum_{i=M+2}^{N} C_i - M \sum_{i=M+2}^{N} \lambda_i \right) t_M$$

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

$$\left| \bigcup_{i=0}^{M} R_{i,M+1}(t_M, t_{M+1}) \right| \le (t_{M+1} - t_M) \sum_{i=0}^{M} C_i. \tag{21}$$

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

$$\left| \bigcup_{i=M+2}^{N} R_{i,M+1}(t_M, t_{M+1}) \right| \le \min \Bigg(\sum_{i=M+2}^{N} C_i (t_{M+1} - t_M), \sum_{i=M+2}^{N} \lambda_i t_1 - \left[\sum_{i=M+2}^{N} C_i - \sum_{i=M+2}^{N} \lambda_i M \right] t_M \Bigg). \tag{22}$$

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} shown in (23). Also note that by definition, $F_{M+1}(t_{M+1}) = F$.

Considering each term of the min in (23) 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 $\sum_{i=M+2}^{N} C_i t_{M+1} \le \sum_{i=M+2}^{N} (M+1) \lambda_i t_1$,

$$\underline{t}_{M+1}(C - C_{M+1}) \ge \underline{t}_M (M-1) \sum_{i=1}^{M} \lambda_i - \underline{t}_M \lambda_{M+1} + \underline{t}_M C_0 + \underline{t}_M M \sum_{i=M+2}^{N} \lambda_i + |F| \tag{24}$$

and in the converse case

$$\underline{t}_{M+1} \sum_{i=0}^{M} C_i \ge \underline{t}_M (M-1) \sum_{i=1}^{M} \lambda_i - \underline{t}_M \lambda_{M+1} + \underline{t}_M C_0 - \underline{t}_M \sum_{i=M+2}^{N} \lambda_i + |F|. \tag{25}$$

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

$$\lambda_{M+1} = \frac{C_{M+1}}{M}. \tag{26}$$

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

$$
\begin{aligned}
|F_{M+1}(t_{M+1})| \quad \leq \quad & \left(\sum_{i=1}^{M} C_i - (M-1)\sum_{i=1}^{M}\lambda_i\right) t_M - M\sum_{i=M+2}^{N}\lambda_i t_M + \lambda_{M+1}t_M \\
& +(t_{M+1}-t_M)\left(C_0 + \sum_{i=1}^{M} C_i\right) + \min\left(\sum_{i=M+2}^{N} C_i t_{M+1},\ \sum_{i=M+2}^{N}(M+1)\lambda_i t_1\right)
\end{aligned} \tag{23}
$$

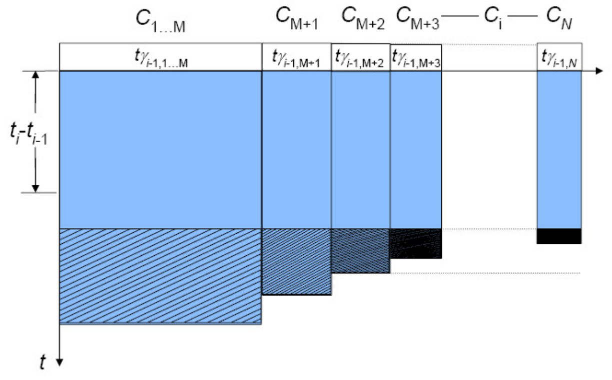

Fig. 6. A visual depiction of the waterfilling argument for the case when $1 < M < N$ and $C_0 > C_0^*$. Note the tiered structure of the columns for $i > M$.

Since the bound given by (24) is increasing in $\sum_{i=M+2}^{N}\lambda_i$ and that given by (25) is decreasing, the min in (23) is minimized, for a given $C_0 = \sum_{i=1}^{N}\lambda_i$, when the two bounds coincide. This gives the fundamental lower bound

$$
\underline{t}_{M+1} \geq \frac{(M^2C_0 - M^2\lambda_{M+1} + 2MC_0 - M\lambda_{M+1} + C_0)|F|}{C_0\left((M+1)(\sum_{i=0}^{M} C_i) + M\sum_{i=M+2}^{N} C_i\right)}. \tag{27}
$$

When $C_0 > C_0^*$, the value of $\sum_{i=1}^{M}\lambda_i$ necessary to achieve this bound violates (19). In this case, the algorithm sets λ_i, $i < M$, to its upper bound of $C_i/(M-1)$, and (22) becomes $|\bigcup_{i=M+2}^{N} R_{i,M+1}(t_M, t_{M+1})| = \sum_{i=M+2}^{N} C_i t_{M+1}$.

When $C_0 > C_0^*$, nodes $i \in \{M+2, \ldots, N\}$ need not upload *all* of their information $F_i(t_M)$ to node M to achieve the lower bound (22); it is sufficient that λ_i, $i \in \{M+2, \ldots, N\}$, be large enough that $r_{i,M+2}(t) = C_i$ for all $t \in (t_M, t_{M+1})$. The LP (5) ensures that condition is met, while sequentially providing as much server capacity on $(0, t_M)$ as possible to nodes $M+2, \ldots, N$.

In either case, Algorithm 1 achieves the lower bound on $\underline{t}_{M+1}$ while maintaining $t_1, \ldots, t_M = |F|/C_0$.

Finally, we claim after $\underline{t}_{M+1}$, 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 $k \notin \{1, \ldots, N\}$, needs to receive all information held by nodes $\{1, \ldots, N\}$ (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 Algorithm 1,

$$
\lambda_N \leq \lambda_i, \quad \forall i \in \{1, \ldots, N\}, \tag{28}
$$

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

$$
\frac{\sum_{i=1}^{M}\lambda_i}{C_0 + \sum_{i=1}^{N} C_i} \leq \frac{\lambda_N}{C_N} \tag{29}
$$

which can be reformed as

$$
\frac{C_0}{C} \leq \frac{\sum_{i=M+2}^{N}\lambda_i}{\sum_{i=M+2}^{N} C_i} \tag{30}
$$

and results in a bound of

$$
C_0 \geq \frac{C_{M+1}(C - C_0)}{MC_{M+1} + \sum_{i=M+2}^{N} C_i}. \tag{31}
$$

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. ■

Appendix B
Argument of Claim 2 with $C_0 > C_0^*$

The maximum amount of information which can go into set A by time t_{M+1} is

$$
\begin{aligned}
&\left(C_0 + \sum_{i=1}^{M} C_i\right) t_{M+1} + C_{M+1}t_M - \sum_{i=M+2}^{N} F_i(t_{M+1}) \\
&+ \min(t_{M+1}\sum_{i=M+2}^{N} C_i, (M+1)\sum_{i=M+2}^{N} F_i(t_{M+1}))
\end{aligned} \tag{32}
$$

As shown earlier, this expression is maximized with respect to $\sum_{i=M+2}^{N} F_i(t_{M+1})$ for any t_M and t_{M+1} when

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

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

$$
t_{M+1} \geq \frac{(M+1)|F| - C_{M+1}t_M + \epsilon}{C - C_{M+1} - \sum_{i=M+2}^{N}\frac{C_i}{M+1}}. \tag{33}
$$

When $\sum_{i=1}^{M} t_i$ is added to both sides in (33), it becomes clear that the relationship between ϵ and $t_1 \ldots t_M$ is key to the minimization of the sum $\sum_{i=1}^{M+1} t_i$. In particular, we look to construct a function $f(\epsilon)$ such that

$$\sum_{i=1}^{M} t_i \geq f(\epsilon). \tag{34}$$

Let $B = (C_0 - C_0^*)|F|/C_0$ be the excess data which the seed can send in bottleneck time using capacity above C_0^*. Since ϵ corresponds to extra capacity given to nodes $M+2, \ldots, N$, we can consider $B - \epsilon$ to be the extra data given to nodes $1, \ldots, M+1$ from what remains of this excess capacity.

From the multiplicity theorem, without considering the effect of ϵ, we know that a tight bound on $\sum_{i=1}^{M} t_i$ is $M|F|/C_0$. Now, we maintain

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

(It can be assumed $\sum_{i=M+2}^{N} F_i(t_M) = \sum_{i=M+2}^{N} F_i(t_{M+1})$, since any change in the content of nodes $M+2, \ldots, N$ after time t_M has no benefit to finish time t_{M+1}). For $\epsilon < B$, it follows that $B - \epsilon$ capacity must be absorbed directly from the server by nodes $1, \ldots, M+1$, since the $C_0 = C_0^*$ scheme has all data for $\sum_{i=M+2}^{N} F_i(t_{M+1})$ coming directly from the server.

This means that nodes $1, \ldots, M+1$ will be oversaturated, such that $\lambda_i > C_i/(M-1)$ for at least one node in the set $1, \ldots, M$, or $\lambda_{M+1} > C_{M+1}/M$. As a direct result, $t_M > |F|/C_0$, since each of these nodes cannot send to all M nodes. The amount of information which will remain to be sent into the set will be at least $B - \epsilon$, and can only be held by members of the set. The amount of time to send this information to nodes in the set which remain to finish will be at least $(B - \epsilon)/\sum_{i=0}^{M-1} C_i$. Let

$$f(\epsilon, k) = \frac{(B-\epsilon)}{\sum_{i=0}^{M-1} C_i} + \frac{k|F|}{C_0}.$$

Then (34) is satisfied by $f(\epsilon) = f(\epsilon, M)$. Substituting this lower bound for $\sum_{i=1}^{M} t_i$ into (33) (with $\sum_{i=1}^{M} t_i$ added to both sides), by considering the worst-case scenario of $t_M = f(\epsilon, 1)$, yields a lower bound on $\sum_{i=1}^{M+1} t_i$ in terms of ϵ. The derivative with respect to ϵ,

$$\frac{1}{\sum_{i=0}^{M-1} C_i + C_{M+1}} \left(\frac{C_{M+1}}{\sum_{i=0}^{M} C_i + \frac{M}{M+1} \sum_{i=M+2}^{N} C_i} - 1 \right) + \frac{1}{\sum_{i=0}^{M} C_i + \frac{M}{M+1} \sum_{i=M+2}^{N} C_i},$$

is negative since

$$C_{M+1} + \sum_{i=0}^{M-1} C_i + C_{M+1} - \left(\sum_{i=0}^{M} C_i + \frac{M}{M+1} \sum_{i=M+2}^{N} C_i \right)$$
$$= 2C_{M+1} - \left(C_M + \frac{M}{M+1} \sum_{i=M+2}^{N} C_i \right) < 0.$$

The remainder of the $C_0 \leq C_0^*$ case holds.

References

[1] J. Chan, V. Li and K. Lui. Performance comparision of scheduling algorithms for peer-to-peer collaborative file distribution. *IEEE Journal on Selected Areas in Communications*, 25(1):146–154, January 2007

[2] P. Chou and Y. Wu. Network coding for the Internet and wireless networks. *IEEE Signal Processing Magazine*, 24(5):77–85, September 2007

[3] B. Fan, D. Chiu and J. Lui. The delicate tradeoffs in Bit Torrent-like file sharing protocol design. In *Proccedings of IEEE ICNP*, 2006.

[4] L. Gao, D. Towsley and J. Kurose. Efficient schemes for broadcasting popular videos. In *Proceedings of ACM NOSSDAV*, 1998.

[5] 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.

[6] 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

[7] T. Ho, M. Mèdard and R. Koetter. An information-theoretic view of network management. *IEEE Transactions on Information Theory*, 51(4):1295–1312, April 2005

[8] R. Kumar and K. Ross. Peer-assisted file distribution: The minimal distribution time. In *Proccedings of IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2006.

[9] J. Kurose and K. Ross. *Computer Networking*. Fourth edition, Addison Wesley, 2007.

[10] S. Li, R. Yeung and N. Cai Linear network coding *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003

[11] 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.

[12] S. Liu, R. Shen, W. Jiang, J. Rexford and M. chiang. Performance bounds for peer-assisted live streaming. *Proceedings of ACM SIGMETRICS*, 2008.

[13] L. Massoulié and M. Vojnović. Coupon Replication Systems. *IEEE/ACM Transactions on Networking*, 16(3):603–616, June 2008

[14] M. Mehyar. Distributed Averaging and Efficient File Sharing on Peer-to-Peer Networks. Doctoral Thesis, California Institute of Technology, 2006.

[15] 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.

[16] J. Mundinger, 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.

[17] J. Mundinger, R. Weber and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. *Journal of Scheduling*, 11(2):1094-6136, April 2008.

[18] Q. Ou and D. Tsang. An optimal bandwidth allocation algorithm for file distribution network. In *Proceedings of ChinaCom*, 2007.

[19] 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.

[20] D. Qiu, and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *Proceedings of ACM Sigcomm*, 2004.

[21] S. Sanghavi, B. Hajek and L. Massouliè. Gossiping with multiple messages. *IEEE Transactions on Information Theory*, 53(12):4640–4654, December 2007

[22] 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.

[23] X. Yang and G. De Veciana. Service capacity of peer to peer networks *Proceedings of IEEE Infocom*, 2004.

[24] 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.

[25] X. Zheng, C. Cho and Y. Xia Optimal peer-to-peer techniques for massive content distribution. In *Proceedings of IEEE Infocom*, 2008.