Issues in Peer-to-Peer Networking: a Coding Optimization Approach

Christopher S. Chang*, Tracey Ho*, Michelle Effros*, Muriel Médard†, and Ben Leong‡

* Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA
† Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
‡ School of Computing, National University of Singapore, Singapore 117417, Republic of Singapore
E-mail: {cswchang, tho, effros}@caltech.edu, medard@mit.edu, benleong@comp.nus.edu.sg

***Abstract*—In this paper we consider a linear optimization approach for studying download finish times in peer-to-peer networks that allow but do not require coding. We demonstrate that using the network coding framework simplifies analysis even in scenarios where the optimal solution does not require coding. For example, we use the network coding framework to disprove the claim of Ezovski *et al.* that in the absence of coding, the sequential minimization of file download times minimizes the average finish time over all users. We also use this framework to study the effect of requiring reciprocity, a typical feature of incentive-compatible protocols. Lastly, we show that for a dynamically changing network scenario, coding can provide a robust and optimal solution that outperforms routing.**

I. Introduction

Peer-to-peer (P2P) file distribution algorithms are an active field of research in both academic [1] and industrial [2] settings. P2P algorithms are desirable for their scalability—allowing efficient and inexpensive file distribution from a single content provider (server) to thousands of users. P2P systems achieve these benefits by exploiting the bandwidth of the peers.

While network coding has been applied to P2P systems to improve robustness and maximize throughput [3], the performance gain of network coding over routing in a P2P system remains a topic for further research. In [4], Deb *et al.* consider the dissemination of multiple messages using a gossip based protocol, showing that in an n-node network, network coding speeds message dissemination from time $\Theta(n \log(n))$ for uncoded schemes to time $O(n)$ for random linear coding. In [5], Mundinger and Weber introduce an uplink sharing model that assumes a fully connected network where each peer is constrained only in its upload capacity. In [6], Chiu *et al.* show that network coding does not increase multicast throughput in this scenario. In [7], Mehyar *et al.* investigate a few small networks of this type, studying optimal strategies for minimizing (a) the finish time of the last peer; (b) the average finish time over all users; and (c) the Min-Min finish time, which sequentially minimizes the finish time of the remaining peer with the highest upload capacity until all peers finish their downloads. In [8], Ezovski *et al.* present an optimal routing solution for minimizing the Min-Min finish time in the uplink sharing model. They further claim that following the Min-Min strategy minimizes the average finish time over all routing strategies.

In Section IV, we use a counterexample to disprove Ezovski *et al.*'s claim that the Min-Min strategy minimizes the average finish time. Like [7], [8], we assume that all peers stay in the system after completing their own downloads. We derive our counterexample using the linear programming approach of Wu *et al.* from [9]. We also extend the LP to study the effect of a reciprocity constraint. Finally, we show that coding can improve robustness to unexpected network changes.

Our investigations underscore two benefits of the network coding framework. First, the framework makes the problem of finding optimal solutions tractable; even when routing suffices to obtain the optimal performance, finding the optimal routing solution is often an intractable problem. Second, coding can provide robustness in dynamically changing network scenarios.

II. Preliminaries

We use the uplink sharing model of [5]. The network is fully connected, and the upload capacity of each node (including all peers and the server) is initially the only constraint. We discuss the system performance in terms of download finish times when there is a single server with a finite file to distribute to multiple peers. We consider the following performance metrics:

1) *Min-Min Finish Time:* The Min-Min finish time strategy sequentially minimizes the finish time of the remaining node with the highest capacity. Precisely, let T_i be the finish time for the i^{th} node when nodes are ordered from largest to smallest upload capacity. Then the Min-Min strategy is the strategy that achieves

$$\begin{aligned} T_1^* &\triangleq \min T_1 = \frac{(\text{File Size})}{(\text{Server Upload Capacity})} \\ T_i^* &\triangleq \min\{T_i | T_j = T_j^*, \forall j < i\} \end{aligned} \quad (1)$$

The optimization is over all possible routing or coding schedules that satisfy the nodes' uplink capacity constraints.

This work has been supported by the Lincoln Lab from AFRL contract no FA8721-05-C-0002 and by Singapore Ministry of Education grant R-252-000-348-112.

2) *Min-Avg Finish Time:* The Min-Avg finish time strategy minimizes the average finish time over all users. If peers stay in the network until all downloads are completed, then peers finish in the order of highest to lowest upload capacity as in the Min-Min strategy.

Both metrics can be applied either with or without reciprocity constraints.

III. Linear Programming Formulation

Let v_0 and $v_1, \ldots, v_m$ denote the server and peers respectively in a single-server, m-peer P2P network. Node $v \in \{v_0, \ldots, v_m\}$ has uplink capacity $c(v)$. All peers remain in the network after finishing their downloads. It is therefore always optimal for higher capacity peers to finish earlier than lower capacity peers since their greater upload capacity makes them more useful for serving other peers. We therefore order the peers from highest to lowest upload capacity giving $c(v_1) \geq c(v_2) \geq \cdots \geq c(v_m)$. We describe each solution for distributing a file of size F from the server to the peers by describing a sequence of phases. Each phase is a period in which the upload strategies of all nodes are fixed—that is, in phase τ each node $v \in \{v_0, \ldots, v_m\}$ allocates its upload capacity according to some fixed flow vector describing the proportion of node v's upload capacity used to upload data to each of the users in $\{v_1, \ldots, v_m\} \setminus \{v\}$. The duration of phase τ equals the maximum over nodes $v \in \{v_0, \ldots, v_m\}$ of the total flow from node v in phase τ divided by the uplink capacity of node v. To make this precise, we represent a full solution with I phases by the following time-expanded graph. Let $\mathcal{V} = \{v_0^{(\tau)}, v_1^{(\tau)}, \ldots, v_m^{(\tau)}\}_{\tau=1}^{I}$, where $v_i^{(\tau)}$ represents node v_i in phase $\tau \in \mathcal{I} \triangleq \{1, \ldots, I\}$. Let $\mathcal{E}$ denote the set of edges in the time-expanded graph. Set $\mathcal{E}$ contains two types of edges:

- Transmission edge $e = (v_i^{(\tau)}, v_{i'}^{(\tau)})$ corresponds to the transmission from v_i to $v_{i'}$ within the τ^{th} phase.
- Memory edge $e = (v_i^{(\tau)}, v_i^{(\tau+1)})$ corresponds to the accumulation of received information from previous time steps. Memory edges have infinite capacities.

Each transmission edge exists within a single phase. Each memory edge crosses from one phase to the next. As in [7], [8], we assume continuous data flow and allow each node to forward data immediately upon receipt[1]. Further, each node can transmit data to multiple nodes simultaneously.

Since there always exists an optimal strategy with $I \leq m$ [10], we set $I = m$ and treat $v_\tau^{(\tau)}$, $\tau \in \{1, \ldots, m\}$ as the sink nodes of the time-expanded graph. Let $\mathbf{t} = (t_1, \ldots, t_m)$ denote the vector of phase durations. Peer j finishes its download in the j^{th} phase, so its finish time is $T_j = \sum_{k=1}^{j} t_k$. Both the Min-Avg and Min-Min objective functions can be described as linear functions of vector $\mathbf{t}$. For Min-Avg, let $\mathbf{d} = \left[\frac{m}{m}, \frac{m-1}{m}, \ldots, \frac{1}{m}\right]$. Then the Min-Avg objective function is

$$\mathbf{d}^T\mathbf{t} = \frac{1}{m}\sum_{j=1}^{m}\left[\sum_{k=1}^{j} t_k\right].$$

To represent Min-Min as a linear function of $\mathbf{t}$, we introduce a weighting vector ω such that $\omega_1 \gg \omega_2 \gg \ldots \gg \omega_m$ and set $\mathbf{d} = \left[\frac{\omega_1+\cdots+\omega_m}{m}, \frac{\omega_2+\cdots+\omega_m}{m}, \ldots, \frac{\omega_m}{m}\right]$. The Min-Min objective function is

$$\mathbf{d}^T\mathbf{t} = \frac{1}{m}\sum_{j=1}^{m}\omega_j\left[\sum_{k=1}^{j} t_k\right].$$

Given this framework, the search for the most efficient download strategy becomes a linear programming (LP) problem, as described in [9] and restated below. In this LP, the optimization variables $f_i(e)$ and $x(e)$ represent the virtual flow to node i through edge $e \in \mathcal{E}$ and the total flow through edge $e \in \mathcal{E}$, respectively. The virtual flow $f_i(e)$ is the flow over edge e that is useful to node i. The LP is then given by

$$\begin{aligned}
\min_{\mathbf{f},\mathbf{x},\mathbf{t}} \quad & \mathbf{d}^T\mathbf{t} \\
s.t. \quad & x(e) \geq f_j(e),\ \forall j \in \{1,\ldots,m\},\ \forall e \in \mathcal{E} \\
& \sum_{v_j^{(\tau)}} \frac{x((v_i^{(\tau)}, v_j^{(\tau)}))}{c(v_i)} \leq t_\tau,\ \forall v_i^{(\tau)} \in \mathcal{V},\ \forall \tau \leq I \\
& f_j(e) \geq 0, \quad \forall j \in \{1,\ldots,m\},\ \forall e \in \mathcal{E} \\
& \sum_{v_k^{(\tau)}} f_j((v_k^{(\tau)}, v_i^{(\tau)})) - \sum_{v_k^{(\tau)}} f_j((v_i^{(\tau)}, v_k^{(\tau)})) \\
& \quad = \left\{ \begin{array}{ll} F & \text{if } \tau = i \\ -F & \text{if } i = 0,\ \tau = 1 \\ 0 & \text{otherwise} \end{array} \right\}, \forall j \in \{1,\ldots,m\}
\end{aligned} \tag{2}$$

The first constraint sets the total flow on each edge to the maximum among all virtual flows over the edge; this value suffices for multicast network coding by [11]. The second constraint requires that the duration of each phase be the maximum, over all nodes, of the time required for node v to deliver its flow for the given phase; we calculate this value as the total flow out of node v divided by the upload capacity of v. The third constraint requires that all flows be non-negative. The last constraint guarantees the conservation of flow at all nodes in the network.

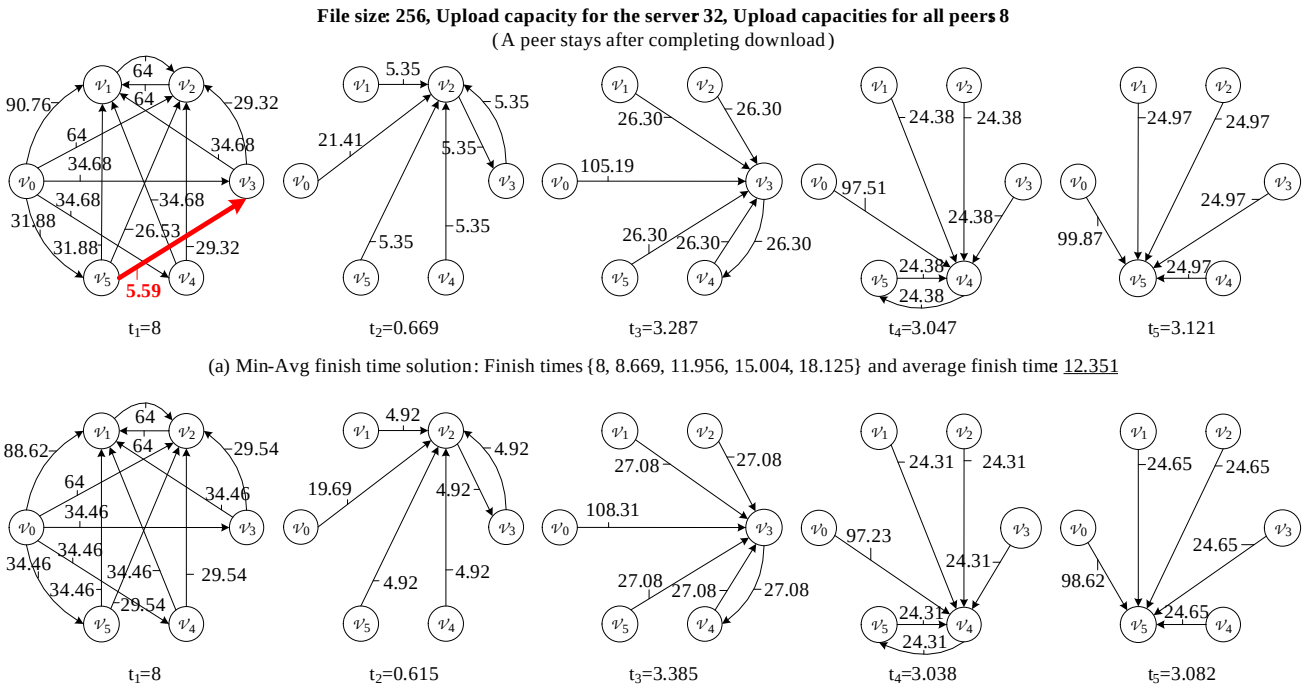
In [9], the LP is stated without an explicit proof. We provide a proof in the extended version of this paper [10].

IV. Min-Min vs. Min-Avg Finish Times

In this section, we show by example that routing algorithms achieving Min-Min finish times do not necessarily minimize the average finish time. This contradicts Claim 1 in [8].

A counterexample with five peers is shown in Fig. 1. The source has upload capacity 32, and each peer has upload capacity 8. We use the LP (2) to find optimal flow solutions for Min-Avg and Min-Min. We then show that routing is sufficient to achieve the optimal solutions in both cases. We prove the existence of an optimal routing solution by explicitly labeling the identities of the flows (see Fig. 2). The labeling procedure

[1]This simplifying assumption is not realistic in practice since nodes typically cannot send out data until they receive at least a block of a certain size. As a result, optimal download times achieved using this model give lower bounds on the download times that can be achieved in practice.



(a) Min-Avg finish time solution: Finish times {8, 8.669, 11.956, 15.004, 18.125} and average finish time 12.351

(b) Min-Min finish time solution: Finish times {8, 8.615, 12, 15.039, 18.120} and average finish time: 12.355

Fig. 1. Optimal flow solutions for a P2P network with 5 peers and upload capacity constraints $c(0) = 32$ and $c(i) = 8$, $i \in \{1, \ldots, 5\}$. Each graph shows a single phase. Edges are labeled with the total amount of flow along the edge in that phase. Recall that t_i denotes the duration of the i^{th} phase. Min-Avg scheduling (a) has smaller average finish time than Min-Min scheduling (b). The main difference comes from the bold link in red.

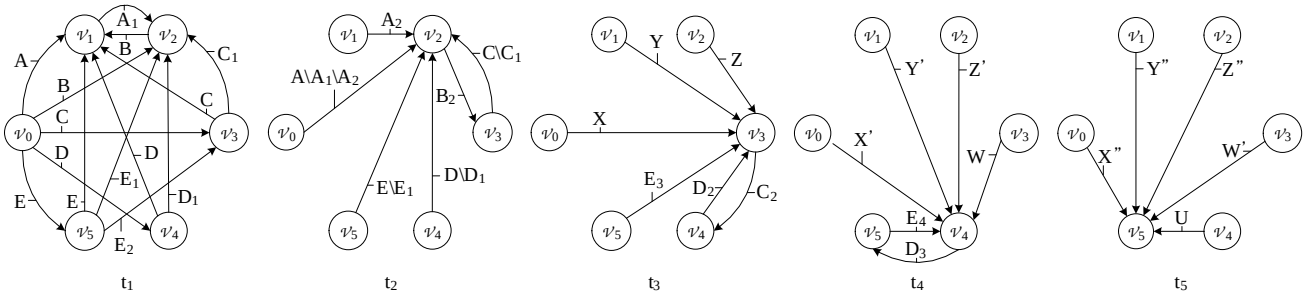


Fig. 2. A routing realization of the optimal network coding solution from Fig. 1. Each edge carries the same amount of information as the corresponding edge from Fig. 1. Edges are labeled with the identity of the information being transmitted. The correctness of the first two phases is easy to verify. In the later phases, $C_2 \subseteq C$, $D_2 \subseteq D$, $D_3 \subseteq D$, $E_3 \subseteq E \setminus E_2$, $E_4 \subseteq E$ can be any subsets satisfying causality. The other data flows X, Y, Z, U, W are from nodes that have the entire file.

is simplified by noting that by the i^{th} time step, the first i peers each have all of the data. As a result, for the $(i+1)^{th}$ phase, we only need to check if peers $i+2, i+3, \ldots, m$ can send distinct data to peer $i+1$. In this example, the average finish time for Min-Avg is less than that of the Min-Min solution, which contradicts Claim 1 in [8]. Let M be the maximal number of users that can finish in the "*bottleneck time*," $F/c(0)$. In [8], the authors tried to prove Claim 1 by first showing that it is necessary to minimize $\sum_{i=1}^{M+1} T_i$ in order to minimize the average finish time $\sum_{i=1}^{m} T_i$. However, the counterexample in Fig. 1 shows that minimizing $\sum_{i=1}^{M+1} T_i$ is *not* necessary for minimizing $\sum_{i=1}^{m} T_i$.

In the example of Fig. 1, the main difference between the two strategies occurs in the first phase. For Min-Min scheduling, peer 5 sends data to peers 1 and 2 only. For Min-Avg scheduling, peer 5 sends data to peers 1, 2, and 3 (the bold link in red in Fig. 1(a)). Sending data to peer 3 delays peer 2's finish time in the second phase but significantly reduces the duration of the third phase.

We observe empirically that the Min-Avg and Min-Min strategies can differ for networks with more than 4 peers. For most randomly generated capacity values, the difference between the finish times resulting from the two strategies is nonzero but small. When the peers all have the same upload capacities, the gap increases with the number of peers. The difference between the finish times of Min-Avg and Min-Min is 0.032% for the 5 peer example in (Fig. 1), and 0.171% for an example with 10 peers, all with capacity constraint 8.

This example illustrates the power of the network coding framework for routing problems. Finding an optimal routing solution directly is often extremely difficult. Using the given LP, we can find an optimal coding solution in polynomial time. In some cases, applying our labeling strategy to the resulting coding solution allows us to demonstrate that the optimal solution is also achievable with routing alone.

V. Reciprocity Constraints

Reciprocity is a concept used in incentive-compatible protocols to encourage users to operate in a manner that benefits the entire network. In this section, we show how the LP approach can be used to study the effect of reciprocity. The goal of reciprocity constraints is to encourage peers joining the network to help in distributing information to other users. A number of simple models for reciprocity are possible. For example, one model of reciprocity sets a reciprocity constant $\rho \in [0, 1]$ and imposes the constraint that v_i should send to v_j approximately the same amount of information as v_j sends to v_i in each phase—more precisely, the two flows should differ by at most a factor ρ. The reciprocity constraint can be applied to virtual flows as

$$\rho f_i(v_j^{(\tau)}, v_i^{(\tau)}) \leq f_j(v_i^{(\tau)}, v_j^{(\tau)}) \leq f_i(v_j^{(\tau)}, v_i^{(\tau)})$$

or to actual flows as

$$\rho x(v_j^{(\tau)}, v_i^{(\tau)}) \leq x(v_i^{(\tau)}, v_j^{(\tau)}) \leq x(v_j^{(\tau)}, v_i^{(\tau)}).$$

Lemma 1 shows that the two definitions are equivalent since $f_j(i, j) = x(i, j)$ for all i, j. A variety of other definitions of reciprocity are possible. One alternative way to model reciprocity is to set a limit on the absolute difference between the cumulative amount sent in each direction. Since both of these constraints are linear, either one can be added to the LP. The examples that follow use the first model.

Lemma 1. *For any P2P file distribution network there exists an optimal solution in which*

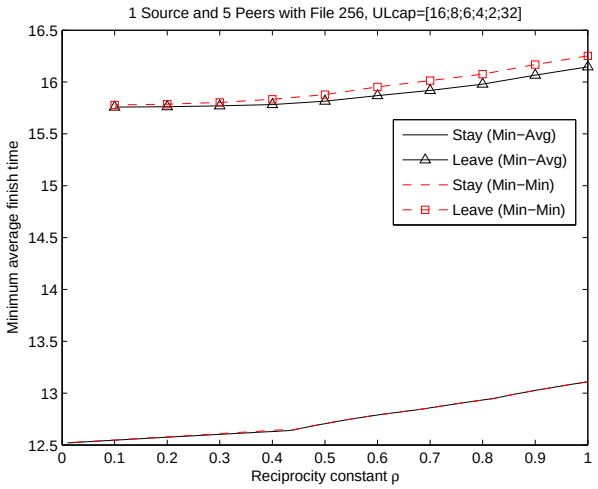
$$f_j(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)}) \quad \textit{for all } (i, j),\ \tau$$

By Lemma 1, definitions of reciprocity in terms of $f_j(i, j)$ and those in terms of $x(i, j)$ are always equivalent.

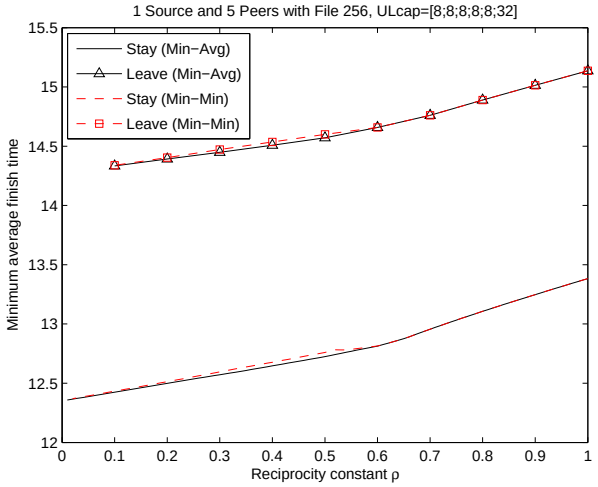
Proof: For a given flow solution, we consider two cases: 1) $f_j(v_i^{(\tau)}, v_j^{(\tau)}) < x(v_i^{(\tau)}, v_j^{(\tau)})$ and 2) $f_j(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)})$. For the first case, we partition the total flow $x(v_i^{(\tau)}, v_j^{(\tau)})$ into the portion that contains the virtual flow $f_j(v_i^{(\tau)}, v_j^{(\tau)})$ and the part that does not. Note that the second part contains information that is linearly dependent on information already received at peer v_j. As a result, node v_j can serve any node v_k that relies on this information without sending this part of the transmission. The following argument makes this precise.

We wish to show that there exists an optimal solution such that $f_j(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)})$ for all i, j. Suppose that an optimal solution has an edge $(v_i^{(\tau)}, v_j^{(\tau)})$ for which $f_j(v_i^{(\tau)}, v_j^{(\tau)}) < x(v_i^{(\tau)}, v_j^{(\tau)})$. Since $x(v_i^{(\tau)}, v_j^{(\tau)}) = \max_k f_k(v_i^{(\tau)}, v_j^{(\tau)})$, there exists some $k \neq j$ for which $f_k(v_i^{(\tau)}, v_j^{(\tau)}) = x(v_i^{(\tau)}, v_j^{(\tau)})$. Since $x(v_i^{(\tau)}, v_j^{(\tau)}) > f_j(v_i^{(\tau)}, v_j^{(\tau)})$, the given solution sends $x(v_i^{(\tau)}, v_j^{(\tau)}) - f_j(v_i^{(\tau)}, v_j^{(\tau)})$ bits from peer v_i to peer v_j for use by peer v_k, but all of these bits are linearly dependent on bits already known to peer v_j. As a result, we can remove these redundant $x(v_i^{(\tau)}, v_j^{(\tau)}) - f_j(v_i^{(\tau)}, v_j^{(\tau)})$ linearly dependent bits from $v_i^{(\tau)}$ to $v_j^{(\tau)}$ leaving the rest of the solution unchanged. ■

When considering incentive-compatible mechanisms such as reciprocity where non-altruistic peers leave upon completion, there is an inherent design tension between optimizing individual and average finish times, in that average finish time

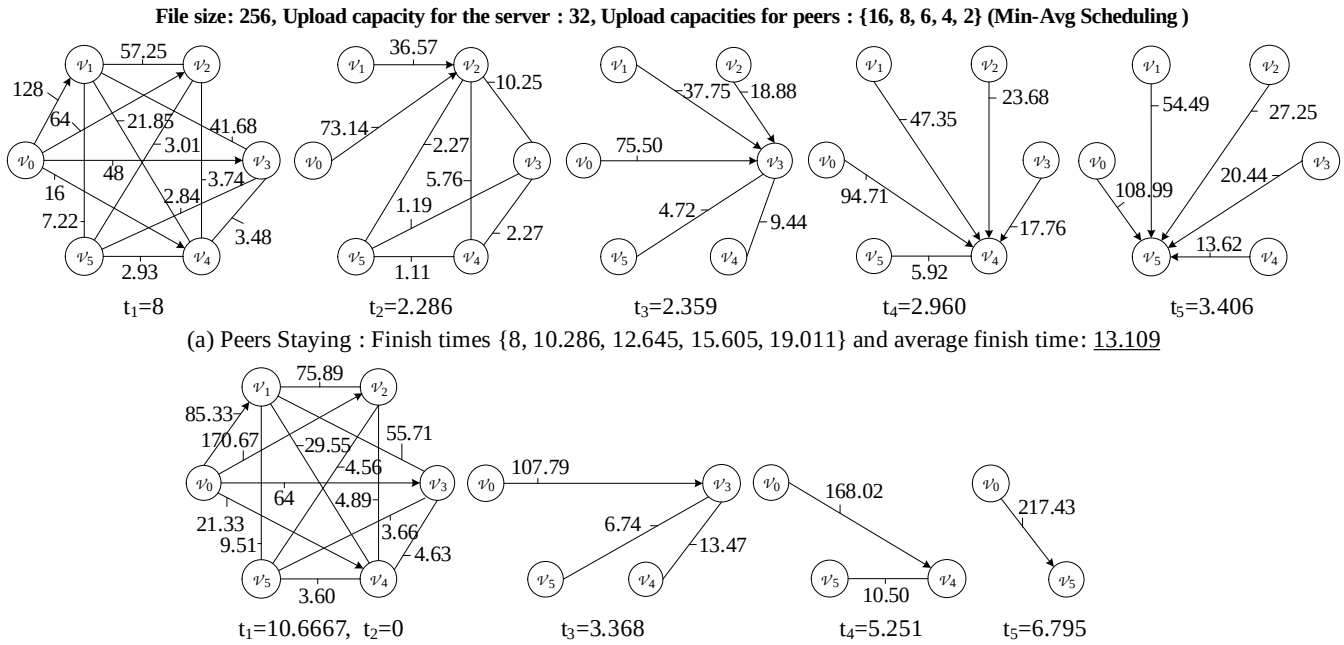


(a) Heterogeneous Capacities



(b) Homogeneous Capacities

Fig. 3. The minimum average finish time versus reciprocity constant ρ. Recall that $\rho = 0$ means no reciprocity, and $\rho = 1$ means a strict reciprocity. The graphs plot average finish times of Min-Avg and Min-Min scheduling for (a) heterogeneous (Upload capacities: server 32 and peers $\{16, 8, 6, 4, 2\}$) and (b) homogeneous (Upload capacities: server 32 and peers 8) upload capacities.



(a) Peers Staying : Finish times {8, 10.286, 12.645, 15.605, 19.011} and average finish time: 13.109

(b) Peers Leaving : Finish times {10.667, 10.667, 14.035, 19.286, 26.081} and average finish time: 16.147

Fig. 4. Optimal Min-Avg flow solutions for a network with heterogeneous capacities case and a strict reciprocity constraint. We consider both (a) the case where peers stay after completing download and (b) the case where peers leave the network after completing download. A link without an arrow denotes a bidirectional link that has the same amount of flow in each direction. Without the reciprocity constraint where peers stay and leave after completing download, the average finish times are 12.517 and 15.756, respectively. Note that peer 1 and 2 finish at the same time in the first phase for (b).

can be improved by delaying the completion time for fast peers so that they continue to contribute upload capacity. We simply note that for any given objective function and ordering of peers finishing, we can use the LP formulation (2) with the following additional linear constraint—requiring all outgoing flows from a peer that completed its download to be zero after its finish time:

$$\sum_{v_j^{(\tau)}} x((v_i^{(\tau)}, v_j^{(\tau)})) \leq c(v_i^{(\tau)}) = 0,\ \forall v_i^{(\tau)} \in \mathcal{V},\ i < \tau \leq I \quad (3)$$

Figure 3 shows Min-Avg and Min-Min finish times for example heterogeneous and homogeneous P2P networks. In both examples the upload capacity of the server is 32. The upload capacities for the peers are $(16, 8, 6, 4, 2)$ in the heterogeneous network and $(8, 8, 8, 8, 8)$ in the homogeneous network. Results for reciprocity coefficients varying from 0 to 1 are included, where $\rho = 0$ means no reciprocity, and $\rho = 1$ means strict reciprocity (i.e., $x(e) = x(e')$ for all pairs). Reciprocity constraints are applied to all nodes except for the server.

As expected, the minimum average finish time increases as ρ increases. We note, however, that in these examples, increasing reciprocity from 0 to 1 increases the minimal average finish times for Min-Min and Min-Avg strategies by less than 10%. In Fig. 4, we show an optimal Min-Avg flow solution for a sample case with a strict reciprocity ($\rho = 1$) and hetherogeneous upload capacities and peers (a) stay or (b) leave the network after completing their downloads.

VI. Robustness Benefit of Coding

In this section, we show that network coding can improve the P2P networks robustness against unforeseen events such as changes in upload capacity, changes in connectivity, or nodes joining or leaving the network unexpectedly.

For instance, consider the following scenario with a file of size 256, a server of capacity 32, and 6 peers whose capacities are $\{16, 8, 8, 8, 8, 8\}$. For the static case without reciprocity, we obtain an optimal flow solution from (2) and find a corresponding routing solution, as described in Section IV (see Fig. 5). Now suppose that the network follows an optimal solution up to the fourth phase, when an unexpected event occurs. By then, the fastest three peers have finished their downloads. As a result, these peers have the complete file, and we treat them as part of an augmented server. This effectively increases the server's capacity from 32 to 64. In the optimal routing solution, $v_5^{(4)}$ and $v_6^{(4)}$ use phase 4 to send to $v_4^{(4)}$ some of the data that v_5 and v_6 received from the server in previous phases. (The relevant flows are highlighted in red in Fig. 5.) The rest of the data is sent to $v_4^{(4)}$ by the server. Therefore, $v_4^{(4)}$ receives directly from the source some of the same information that $v_5^{(4)}$ and $v_6^{(4)}$ have previously received from the server. Before the fourth phase, both v_5 and v_6 receive data only from the server and only in the first phase. In a routing solution, the server can only time share between sending information known to v_5 and information known to v_6 and information known to both. In contrast, in a network coding solution, the

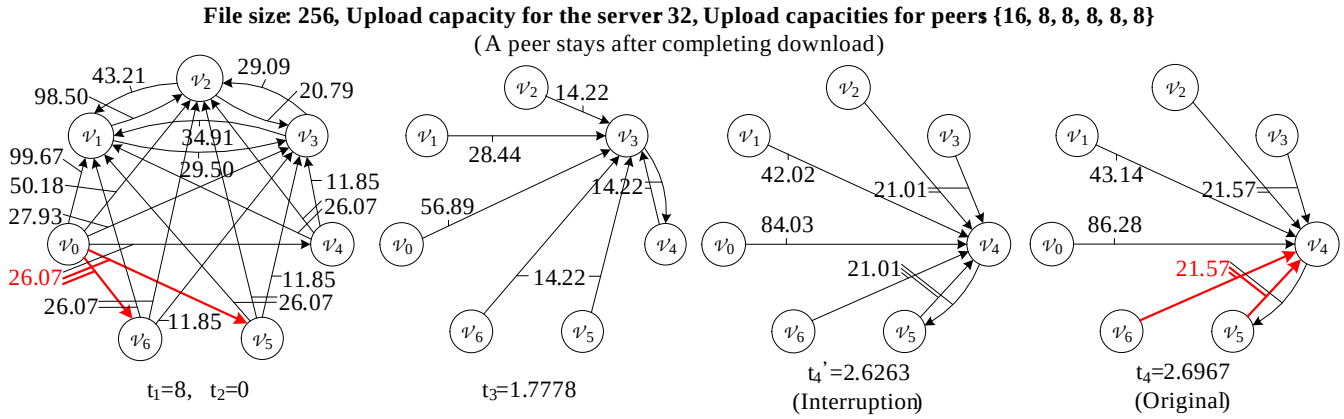


Fig. 5. Optimal flow graph for the example of Section VI for both coding and routing. Note that in this case, v_1 and v_2 finish at the same time in the first phase (the second phase has zero duration). In the static network, the duration of the fourth phase is $2.6967s$. Note that neither v_5 nor v_6 can send the whole data that were received from the server in the first phase (highlighted in red). Instead, the server (including seeds) should compensate by sending the repeated data. Before the end of the fourth phase, at $T = 12.4038s$, an interruption occurs. Until then, v_4 receives the repeated data, the amount of 4.5037.

TABLE I
FINISH TIMES FOR EACH CASE (SERVER CAPACITY IS DECREASED TO 0.1)

	Coding	Routing (half)	Routing (worst)
Finish Time of v_4	18.00	40.52	63.04
Finish Time of v_5	38.19	40.52	63.04
Avg. Finish Time*	16.27	21.25	30.25

* Finish times for the first 3 peers are same for all cases.

server can send linear combinations of these subsets of the data.

Now suppose that at time $\tau \in (T_3, T_4)$, the connectivity between the augmented server and the remaining peers decreases greatly, and, probabilistically, either $v_5^{(4)}$ or $v_6^{(4)}$ leaves the system. Without prior knowledge of which peer will leave the system, any particular choice from the family of time sharing routing solutions will have a worse expected average finish time than the coding solution.

To give a specific numerical example, suppose that the network disruption occurs at 12.4038 seconds (2.6263 seconds after the beginning of the fourth phase). Note that in the fourth phase, the server sends an amount 163.56 of innovative data in the first 2.5559 seconds, and an amount 4.5037 of repeated data in the remaining 0.0704 seconds before the disruption. At the time of the disruption, the capacity of the augmented server decreases to 0.1, and either v_5 or v_6 leaves the network. In the case where v_6 leaves the system, Table I shows the finish times when coding is used, when routing is used and the server sends equal amounts of data known to v_5 and v_6, and when routing is used and the server sends only v_5's data.

Note that the performance gap between coding and routing can be made arbitrarily large by reducing the capacity between the augmented server and the remaining peers after the disruption.

VII. CONCLUSION

In this paper, we apply a linear programming approach based on network coding to analyze download finish times in a P2P network. We disprove the claim in [8] that Min-Min scheduling achieves the minimum average finish time for routing. We also investigate the effect of reciprocity using the LP. Lastly, we show that coding can provide a robust optimal solution, outperforming routing in dynamically changing network scenarios. Ultimately, we expect that the LP can be used to gain insights into how to design practical P2P algorithms, and to predict how different factors will affect the strategies that can be used in practice and the resulting performance.

REFERENCES

[1] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *SIGCOMM Comput. Commun. Rev.* ACM New York, NY, USA, 2004, pp. 367–378.

[2] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer Systems*, vol. 6, 2003.

[3] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," *in Proc. IEEE Infocom*, pp. 2235–45, 2005.

[4] S. Deb, M. Médard, and C. Choute, "Algebraic gossip: A network coding approach to optimal multiple rumor mongering," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2486–2507, 2006.

[5] J. Mundinger and R. Weber, "Efficient file dissemination using peer-to-peer technology," *University of Cambridge, Statistical Laboratory Research Report 2004-01*, 2004.

[6] D. Chiu, R. Yeung, J. Huang, and B. Fan, "Can Network Coding Help in P2P Networks?" *4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2006.

[7] M. Mehyar, W. Gu, S. Low, M. Effros, and T. Ho, "Optimal Strategies for Efficient Peer-to-Peer File Sharing," *in Proc. IEEE ICASSP*, 2007.

[8] G. Ezovski, A. Tang, and L. Andrew, "Minimizing average finish time in p2p networks," *in Proc. IEEE Infocom*, pp. 594–602, 2009.

[9] Y. Wu, Y. Hu, J. Li, and P. Chou, "The delay region for P2P file transfer," *in Proc. IEEE ISIT*, pp. 834–838, 2009.

[10] C. Chang, T. Ho, M. Effros, M. Médard, and B. Leong, "Issues in peer-to-peer networking: a coding optimization approach." [Online]. Available: http://www.its.caltech.edu/~tho/P2P.pdf

[11] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.