Powering Down for Energy Efficient Peer-to-Peer File Distribution

Andrew Sucevic, Lachlan L. H. Andrew, Thuy T. T. Nguyen
CAIA, Swinburne University of Technology, Australia
Email: {landrew, tnguyen}@swin.edu.au

ABSTRACT

Peer to peer (P2P) techniques can reduce the time for file distribution, yet require peers to stay ON to assist others download. We study the total time that peers need to be ON to distribute a file from a server to a set of peers. We show that a P2P system optimized for energy efficiency can consume half the energy of one optimized to minimize download time, with minimal additional delay. To achieve this, peers finish in increasing order of upload capacity, and turn off as soon as they finish downloading. We show that the optimal solution is complex, even for as few as three peers, and advocate heuristics for large networks.

1. INTRODUCTION

Personal computers and their monitors consume a quarter of all ICT energy [2]. This includes many computers being left on to download files, such as software upgrades. This motivates the study of energy-efficient file distribution schemes.

Peer to peer (P2P) technology can reduce the time required for file distribution, but little attention [1,3] has so far been paid to its energy consumption. This paper considers lower bounds on the possible total time that peers need to be active, either uploading or downloading, in order to distribute a file from a single server to a given set of peers. Such bounds are useful both because they give a benchmark against which to compare implementable schemes, and also because they can give structural insights such as indicating the optimal order in which peers should complete.

Following [10,11,15], only upload constraints are considered. It is assumed that a set of peers simultaneously start downloading a given file, and as soon as a peer has downloaded some data, that data can be forwarded to other peers.

In this model, a simple symmetric strategy minimizes the time for the last peer to receive the file [6,11]. This strategy causes all peers to finish simultaneously. However, some peers can finish much earlier with minimal or no increase in the finish time of the last peer. This led to a search for smaller average finish times [5,7,8,14]. The "min-min" problem of sequentially minimizing finish times requires peers to finish in decreasing order of capacity [5]. It was conjectured in [5] that the "min-min" strategy also minimizes the sum of finish times, but a counterexample was presented in [4]. For a given finishing order, [15] derives the polytope of all possible combinations of finish times.

However, these schemes assume that peers will remain on to continue uploading until the last peer has finished downloading. This wastes energy. We instead consider the case that peers may be turned off before the last peer has finished. We show that it is often optimal to turn peers off as soon as they have finished downloading.

A mathematical model of this system is presented in Section 2. For networks of up to three peers, strategies which minimize energy consumption are derived in Section 3 and proven to be optimal. Notably, in these cases peers finish in a different order from the one optimised for delay only, and peers should turn off as soon as they finish downloading. Larger cases are studied by simulation in Section 4, where it is shown that a simple strategy can halve the energy consumption relative to the schemes of [11] and [6].

2. MODEL

We consider a single server distributing a file to N peers. All nodes (server and peers) can communicate with all others, constrained only by the upload capacities. The network is static, in that no peers can arrive or leave. The file is broken up into infinitely small pieces, which allows a peer to forward immediately any data it receives to another peer.

The following notation is used in this model. Peer i can upload any data it has received to any other peers, at rates not exceeding C_i in total. A peer is "finished" when it has received all data in the file. Other notation is:

F: size of the file
N: total number of peers (not including the server)
τ_i: *download time* of the ith peer to finish; $\tau_i \leq \tau_{i+1}$
t_i: *turn off time* of the ith peer to finish; $t_i \geq \tau_i$
C_s: upload capacity of the server
C_j: upload capacity of peer j. Without loss of generality these are in decreasing order: $C_j \geq C_k$ given $j < k$.
p_i: the ith peer to finish. When written as a subscript, this is written pi; for example, C_{p1} is the capacity of the first peer to finish.

We choose rates to minimize the total time peers are on,

$$\sum_{i=1}^{N} t_i. \qquad (1)$$

If the controller does not know the power of each peer, this is the best way to minimize their total energy consumption.

3. OPTIMAL STRATEGIES FOR SMALL N

The strategy which minimizes (1) takes many forms, depending on the relative capacities. In contrast, minimizing the last finish time [6,11] requires only two strategies, while achieving "min-min" finish times [5,10] uses N cases, depending on the "multiplicity" [9], the maximum M such that

$$C_s \le \sum_{i=1}^{M} \frac{C_i}{M-1} + \sum_{i=M+1}^{N} \frac{C_i}{M}. \quad (2)$$

This is the number of peers that can finish at the earliest possible time, F/C_s. When $M = N$, the strategy to minimize (1) matches that of [6,11]: The server sends data to peer i at rate proportional to C_i, which peer i forwards to all other peers. This gives

$$\sum_{i=1}^{N} t_i = \frac{NF}{C_s}. \quad (3)$$

We now describe optimal strategies for cases with up to three peers. As in [5,9], a lower bound on (1) will first be derived by bounding the maximum amount of data that can be provided to a given set of peers in a given time. Then a strategy will be described which achieves this lower bound.

3.1 Networks of 2 Peers

If there are $N = 2$ peers with $C_s \le C_1 + C_2$ then the multiplicity is $M = 2$, and by (3), $t_1 + t_2 \ge \frac{2F}{C_s}$.

When $C_s > C_1 + C_2$ the following Strategy A is optimal.

(i) Select an arbitrary finishing order for the peers.

(ii) During $[0, t_1]$, the server sends to first and second peers at rates $C_s - C_{p2}$ and C_{p2}. Each peer i uploads to the other at rate C_{pi}. Peer p_1 turns off at $t_1 = \tau_1$.

(iii) During $[t_1, t_2]$, the server sends to peer p_2 at rate C_s.

3.2 Network of 3 Peers

When there are three peers, there are at least four cases to consider. The simplest case is again when all $M = N$ can finish by time F/C_s, namely $C_s \le \frac{C_1+C_2+C_3}{2}$. In this case, the minimum cost is $3F/C_s$, by (3).

The case $M = 2$ can be split into two cases: The case when $(C_1 + C_2 + C_3)/2 < C_s < C_1 + C_2$ is still open, and the remaining one is considered in the next section. The difficult case $M = 1$ is presented last.

3.2.1 Multiplicity $M = 2, \frac{C_1+C_2+C_3}{2} < C_1 + C_2 \le C_s$

Consider now the subset of cases with $M = 2$. We will show that the Strategy B below is optimal.

(i) Set the finish order to be (p_1, p_2, p_3) to satisfy: $C_{p2} + C_{p3} \le C_s \le C_{p1} + C_{p2} + C_{p3}/2$

(ii) During $[0, t_1]$: The server sends to peer $i \in \{p_1, p_2\}$ at rate λC_i, where

$$\lambda = \frac{C_s - C_{p3}/2}{C_{p1} + C_{p2}} \quad (4)$$

Peer p_1 sends this data to peer p_2 and vice versa. Both send a subset of this at rate $C_i - \lambda C_i$ to peer p_3.

The server sends to peer p_3 at rate $C_{p3}/2$, which peer p_3 forwards to both peers p_1 and p_2.

Peers p_1 and p_2 turn off at $t_1 = t_2 = \tau_1 = \tau_2$.

(iii) During $[t_1, t_3]$: The server sends to peer p_3 at rate C_s.

Proposition 1. *When $N = 3$ and $C_1 + C_2 < C_s \le C_1 + C_2 + \frac{C_3}{2}$, Strategy B is feasible and minimizes (1). No strategy with $t_2 > F/C_s$ minimizes (1).*

Proof. The full proof is presented in [13]. A sketch follows. The finishing time is bounded below by

$$\sum_{i=1}^{3} t_i \ge \frac{F}{C_s}\left(5 - \frac{C_{p1} + C_{p2} + C_{p3}}{C_s}\right), \quad (5)$$

which can be achieved only if $t_2 = F/C_s$. Under Strategy B,

$$t_1 = t_2 = \tau_1 = \tau_2 = \frac{F}{\lambda C_1 + \lambda C_2 + \frac{C_3}{2}} = \frac{F}{C_s} \quad (6)$$

$$t_3 = \tau_3 = \frac{F}{C_s}(3 - \frac{C_1 + C_2 + C_3}{C_s}). \quad (7)$$

Summing these gives (5) with equality. □

There are again multiple optimal finishing orders; $(p_1, p_2, p_3) = (3, 2, 1)$ is always feasible.

3.2.2 Multiplicity $M = 1, C_1 + C_2 + \frac{C_3}{2} < C_s$

In the optimal strategy, the server sends just enough to p_3 to enable that peer to send continuously to the other two until they finish. Let

$$f(x, y) = 4 - \frac{x}{C_s} + \frac{(2C_s - C_1 - x)(C_s - C_1 - y)}{C_s(C_s + \frac{y}{2})}. \quad (8)$$

and define Strategy C as follows:

(i) If $f(C_2, C_3) < f(C_3, C_2)$, finish in order $(p_1, p_2, p_3) = (2, 1, 3)$; otherwise $(p_1, p_2, p_3) = (3, 1, 2)$.

(ii) During $[0, t_1]$, the server sends different file segments to p_1, p_2 and p_3 at rates $C_s - C_{p2} - r_3$, C_{p2}, and r_3 respectively, where

$$r_3 := \frac{2C_s - C_{p1} - C_{p2}}{2C_s + C_{p3}} C_{p3}. \quad (9)$$

Then p_1 uploads to p_2 at rate C_{p1}, p_2 uploads to p_1 at rate C_{p2}, p_3 uploads to p_1 at rate r_3, and to p_2 at rate $C_{p3} - r_3$. When p_1 obtains the whole file at τ_1, it immediately turns off which means $t_1 = \tau_1$.

(iii) During $[t_1, t_2]$, p_3 continues to upload to p_2 the data it received from the server during $[0, \tau_1]$. The server uploads at full rate to p_2, and p_2 uploads at its full rate to p_3. When p_2 obtains the whole file at τ_2, it immediately turns off which means $t_2 = \tau_2$.

(iv) During $[t_2, t_3]$, p_3 continues to receive the remainder of the file at rate C_s from the server until it obtains the whole file.

The following theorem is proved in Appendix A

Proposition 2. *When $N = 3$ and $C_s > C_1 + C_2 + \frac{C_3}{2}$, under Strategy C, the sum of finish times is*

$$\sum_{i=1}^{3} t_i = \frac{F}{C_s} \min\Big(f(C_2, C_3), f(C_3, C_2)\Big) \quad (10)$$

which is the minimum achievable. Moreover, no strategy in which p_1 turns off after p_2 can achieve this.

4. HEURISTIC FOR LARGE SYSTEMS

The foregoing results show that the optimal strategies quickly become complex as the number of peers grows. However the number of clients downloading a file may be very large. To study this case, it is useful to consider a heuristic strategy based on the insights from the explicit solutions. These insights include: It seems optimal for a peer to turn off as soon as it has finished downloading; The optimal order is not necessarily decreasing order. This gives rise to Strategy D:

(i) Given a finishing order C, calculate the maximum number of peers that can finish at F/C_s,

$$M_C = \min \left\{ M : C_s \leq \sum_{j=1}^{M} \frac{C_{pj}}{M-1} + \sum_{j=M+1}^{N} \frac{C_{pj}}{M} \right\}. \tag{11}$$

(ii) On $[0, F/C_s]$, the server sends to p_i at rate C_{pi}/M_C for $i > M_C$, with the remaining capacity divided among p_1 to p_{M_C} in proportion to C_{pi}. Each peer sends a copy of everything it receives to p_1 to p_{M_C}, except itself. Peers p_1 to p_{M_C} turn off at F/C_s.

(iii) On $[\tau_{i-1}, \tau_i]$, for all $j > i \geq M_C + 1$, p_j sends to p_i at rate C_j; if it only has $D < (\tau_i - \tau_{i-1})C_j$ data, then the server sends it new data at rate $C_j - D/(\tau_i - \tau_{i-1})$ so that p_i can send at rate C_j for the entire time. The server's remaining capacity is sent to p_i. If $i < N$ then p_i sends to p_{i+1} at rate C_{pi}. Peer p_i turns off at $t_i = \tau_i$.

This was evaluated by simulation for varying numbers of peers N, varying distributions of upload capacities C_i, and varying regimes for scaling C_s with N. In each case, a 500 MByte file was distributed to up to 10^4 peers with randomly chosen capacities with mean 10 Mbit/s. The mean of 100 runs of each test is plotted.

Three finishing orders were considered: descending order of capacities, seeking to minimize finish times; random; ascending, to keep the more useful peers active as long as possible. These were compared with two reference strategies: "Simultaneous" [6,11] in which all peers finish simultaneously at the earliest possible time; and "Sequential, No P2P", in which the server sends the complete file to each peer in order, after which the peer turns off. Much less energy would be used if the server could wake each peer when its turn came to download; However, not all computers can be woken remotely, and so a general P2P system cannot rely on that.

4.1 Constant Server Capacity with varying N

Figure 1 shows the objective (1) normalized by the number of peers, for $C_s = 10$ Gbit/s[1] and C_i Pareto distributed, with shape parameter 0.5. As expected, the Sequential scheme increases linearly with the number of peers N, while the P2P schemes scale more gracefully. Even with P2P, the on time increases slightly, since C_s remains fixed even though the peers' total capacity grows in proportion to N. Naively turning peers off while keeping the same finishing order as in [5, 15], in descending order of capacities, increases the energy consumption by a factor of over 2 relative to Simultaneous, since the extra delay incurred by the last peers to finish outweighs the savings. Conversely, finishing the slow peers first and turning them off reduces energy consumption by a factor of about 2 relative to Simultaneous.

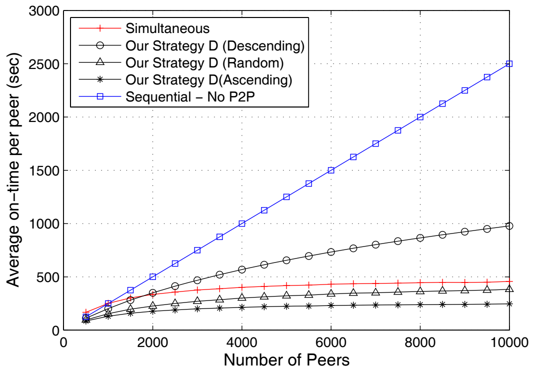

Figure 1: Constant $C_s = 10$ Gbps, C_i is Pareto distributed with mean 10 Mbps.

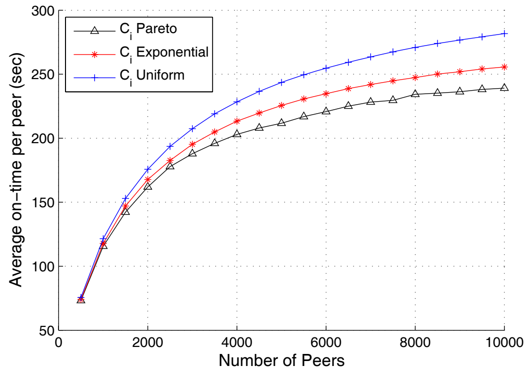

Figure 2: Constant $C_s = 10$ Gbps and different distributions of C_i, with mean 10 Mbps.

The distribution of capacities C_i makes a modest difference to the finish times, as shown in Figure 2 when the C_{pi} are an increasing permutation of C_i. When the C_{pi} are a random permutation, there is no discernible dependence on the distribution. This is presumably because, in that case, the average capacity after i peers have finished remains $10(N - i)$ Mbit/s regardless of the distribution. In contrast, for an increasing permutation, the average capacity decreases sublinearly in i, more slowly as the distribution becomes heavier tailed. This hypothesis is supported by the fact that the heavier tails give *longer* finish times when the C_{pi} are a decreasing permutation.

4.2 Varying Server Capacity

Popular content will typically be served by more powerful servers, and so it is reasonable to expect that the server capacity C_s will increase with the number of peers N. However, it is unclear how it will scale.

One extreme is that C_s may grow proportional to N. Figure 3 shows that this results the server staying on for a constant amount of time, regardless of how many peers there are, since both the amount of data and the capacity scale in proportion. The Strategy D still outperforms the others. The other extreme of constant C_s was shown in Figure 1.

A less extreme scaling would be C_s proportional to $\sqrt{N}$, as shown in Figure 4. In this figure, the capacity with 10^4 peers equals that of Figure 1, and so the only change is an increase in on time for a small number of peers.

[1] Recall that the server is a software house, not a PC.

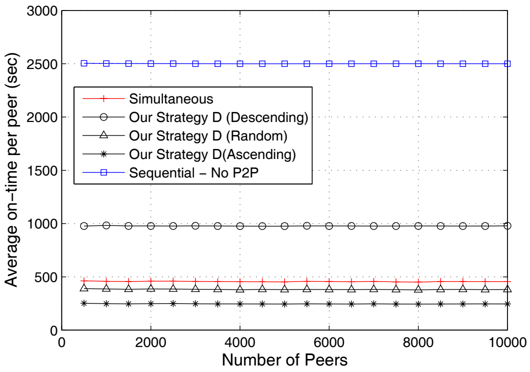

Figure 3: $C_s = 0.1N$ Mbit/s, C_i is Pareto.

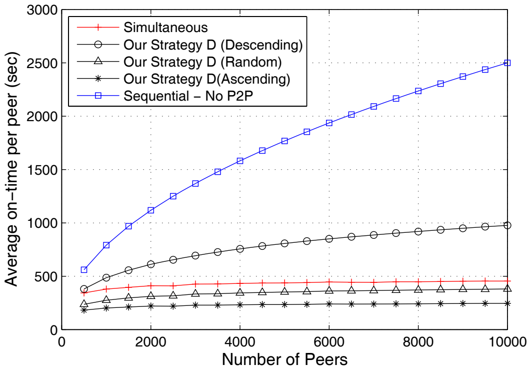

Figure 4: $C_s = 10\sqrt{N}$ Mbit/s, C_i is Pareto.

Figure 5 shows the average on-time per peer using our strategy D as a fraction of the time consumed by the Simultaneously strategy. In all cases of C_s, our strategy takes around half of the time taken using the Simultaneous strategy [2]. This matches the intuition that the ith peer is on for roughly i/N of the time. The upward trend as N increases for constant C_s is because the peers provide more capacity as N increases, and the penalty for turning them off becomes larger. The apparent convergence of all scalings of C_s as N increases is an artefact of the parameters, which make C_s coincide for $N = 10^4$.

5. CONCLUSION

Optimised peer to peer systems can substantially improve the energy efficiency of file distribution compared with systems in which peers are powered on from the time the file becomes available until the time it is downloaded. For small networks, it can be shown rigorously that the optimal strategy turns each peer off as soon as it has finished downloading. It is tempting simply to take a system such as [5] which provides low average download times, and to turn off peers when they finish downloading. However, this actually increases energy consumption. Instead, substantial savings are possible if the order of serving peers is altered so that higher-capacity peers remain in the system longer.

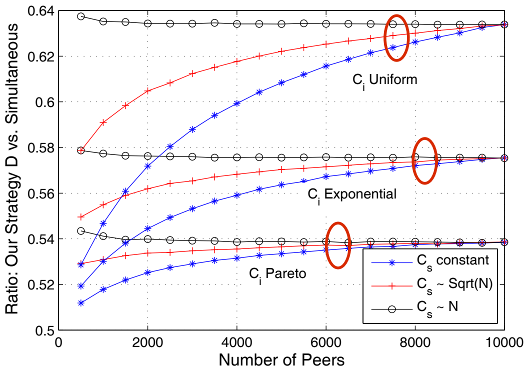

Figure 5: Ratio of Sum of peers' on-time: Our Strategy D vs. Simultaneous. C_{pi} are increasing.

This work considered only an idealized model, but opens the way for many further studies. Apart from the natural tasks of considering download constraints, finding a decentralized solution and considering peers that arrive part way through transmission, which also apply to studies such as [5, 10, 14, 15], there are some extensions specific to energy efficiency. The model could be expanded to include energy consumption of the server, including the non-linear relationship between its energy consumption and capacity. It could also consider peers with known, different power consumptions; in that case, it is likely to be optimal to finish peers in increasing order of upload capacity per watt. Perhaps most importantly, it will be useful to study systems in which peers are not left on purely for P2P downloading; the optimal strategy may be very different when peers are only participating when they are already on for other purposes.

Acknowledgement

This work was supported by Australian Research Council (ARC) grants DP0985322 and FT0991594.

6. REFERENCES

[1] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta. Somniloquy: Augmenting network interfaces to reduce PC energy usage. In *Proc. Usenix NSDI*, 2009.

[2] Australian Computer Society. *Carbon and Computers in Australia*. 2010.

[3] J. Blackburn and K. Christensen. A simulation study of a new green bittorrent. In *ICC Workshops*, 2009.

[4] C. Chang, T. Ho, M. Effros, M. Medard, and B. Leong. Issues in peer-to-peer networking: A coding optimization approach. In *IEEE Int. Symp. Network Coding (NetCod)*, pages 1 –6, June 2010.

[5] G. Ezovski, A. Tang, and L. L. H. Andrew. Minimizing average finish time in P2P networks. In *Proc. IEEE INFOCOM*, 2009.

[6] R. Kumar and K. Ross. Peer-assisted file distribution: The minimum distribution time. In *Proc. IEEE HotWeb*, 2006.

[7] M. Lingjun and K.-S. Lui. Scheduling in P2P file distribution – on reducing the average distribution time. In *IEEE Consumer Commun. Netw. Conf. (CCNC)*, pages 521 –522, Jan. 2008.

[8] M. Lingjun, P.-S. Tsang, and K.-S. Lui. Improving file distribution performance by grouping in peer-to-peer

[2] This comes with a ∼ 10% increase in delay using our strategy D compared to the one which optimised delay.

networks. *IEEE Trans. Netw. Serv. Manag.*, 6(3):149 –162, Sept. 2009.

[9] M. Mehyar. *Distributed Averaging and Efficient File Sharing on Peer-to-Peer Networks*. PhD thesis, California Institute of Technology, 2006.

[10] M. Mehyar, G. WeiHsin, S. H. Low, M. Effros, and T. Ho. Optimal strategies for efficient peer-to-peer file sharing. In *Proc. IEEE ICASSP*, 2007.

[11] J. Mundinger, R. Weber, and G. Weiss. Optimal scheduling of peer-to-peer file dissemination. *J. Scheduling*, 11:105–120, April 2008.

[12] S. Prajna, Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: a general purpose sum of squares programming solver. In *Proc. CDC*, 2001.

[13] A. Sucevic, L. L. H. Andrew, and T. T. T. Nguyen. Minimising peer on-time for energy efficient peer-to-peer file distribution. Online. Available: http://www.caia.swin.edu.au/cv/landrew/pubs/MinOnTime.pdf.

[14] P.-S. Tsang, X. Meng, and K.-S. Lui. A novel grouping strategy for reducing average distribution time in P2P file sharing. In *IEEE Int. Conf. Commun (ICC)*, pages 1 –5, May 2010.

[15] Y. Wu, Y. C. Hu, J. Li, and P. A. Chou. The delay region for P2P file transfer. In *Proc. Int. Symp. Info. Th. (ISIT)*, pages 834–838, 2009.

APPENDIX

A. PROOF OF PROPOSITION 2

First, note that Strategy C is feasible. In particular, since $C_{p3}/2 \le r_3 \le \min\{C_{p3}, C_s - C_{p1} - C_{p2}\}$, the server can allocate rate r_3 to p_3. Next, note that it achieves (10): Since $t_1 = \tau_1 = \frac{F}{C_s}$; $\quad t_2 = \tau_2 = \frac{F}{C_s}\left(\frac{2C_s - C_{p1} - C_{p2}}{C_s + C_{p3}/2}\right)$; $t_3 = \tau_3 = \frac{F}{C_s}(3 + \frac{C_{p1}}{C_s} - \frac{(2C_s - C_{p2})(C_{p1}+C_{p2}+C_{p3})}{C_s(C_s + C_{p3}/2)})$, the sum of on time of peers is

$$\sum_{i=1}^{3} t_i = \frac{F}{C_s}\left(4 - \frac{C_{p1}}{C_s} + \frac{(2C_s - C_{p1} - C_{p2})(C_s - C_{p2} - C_{p3})}{C_s(C_s + C_{p3}/2)}\right). \tag{12}$$

By the choice of the order in which the peers are finished, this is equal to (10). The remainder of this appendix will show that (10) is the optimum, using a series of lemmas established in [13].

We first derive an implicit bound on (1) in terms of the individual finish times. For $i \le N-1$, the ith peer to finish can send at rate at most C_{pi}, and cannot send after t_i. The other peer can send at rate at most $C_{(N)}$, and has no destination left to send to after the second last peer finishes at τ_{N-1}. Finally, the server can send at rate at most C_s, until all peers finish at t_N. Yet all N peers must receive the entire file of size F, whence

$$\sum_{i=1}^{N-1} t_i C_{pi} + \tau_{N-1} C_{(N)} + t_N C_s \ge NF \tag{13}$$

or equivalently

$$t_N \ge \frac{NF - \tau_{N-1}C_{(N)} - \sum_{i=1}^{N-1} t_i C_{pi}}{C_s}. \tag{14}$$

Adding $\sum_{i=1}^{N-1} t_i$ to both sides, and noting that $\tau_N \le t_N$ since peer N must finish before it turns off, gives the lower bound for (1) of

$$\frac{NF + t_{N-1}(C_s - C_{(N-1)} - C_{(N)}) + \sum_{i=1}^{N-2} t_i (C_s - C_{pi})}{C_s}. \tag{15}$$

For $N = 3$, (15) becomes

$$\sum_{i=1}^{3} t_i \ge \frac{3F + t_2(C_s - C_{p2} - C_{p3}) + t_1(C_s - C_{p1})}{C_s}. \tag{16}$$

From the multiplicity theorem, it is impossible that $t_1 = t_2 = \frac{F}{C_s}$ and so (5) is loose. A tight bound will now be derived, using the following lemmas.

Lemma 1. *For $N = 3$ and $M = 1$,*

$$t_2 \ge \frac{2F - C_{p2}\tau_1 - C_{p1}\min\{t_1, t_2\}}{C_s + C_{p3}/2}. \tag{17}$$

Although we know that the download times satisfy $\tau_1 \le \tau_2$, the order of the turn-off times t_i depends on the chosen strategy. Two cases can be considered.

Lemma 2. *For $N = 3$ and $M = 1$, if $t_1 \ge t_2$ then $t_2(C_s - C_{p2} - C_{p3}) + t_1(C_s - C_{p1}) \ge \frac{F}{C_s}(\frac{2C_s(2C_s - C_{p1} - C_{p2} - C_{p3})}{C_s + C_{p1} + C_{p2} + C_{p3}/2})$.*

If $t_1 \le t_2$, then (17) becomes

$$t_2 \ge \frac{2F - C_{p2}\tau_1 - C_{p1}t_1}{C_s + C_{p3}/2}. \tag{18}$$

Multiplying both sides of (18) by $C_s - C_{p2} - C_{p3}$ (which is positive) and then adding $t_1(C_s - C_{p1})$ to both sides gives

$$\begin{aligned} & t_2(C_s - C_{p2} - C_{p3}) + t_1(C_s - C_{p1}) \ge \\ & \frac{(C_s - C_{p2} - C_{p3})(2F - C_{p2}\tau_1)}{C_s + C_{p3}/2} \\ & + \frac{t_1[(C_s - C_{p1})(C_s + C_{p3}/2) - C_{p1}(C_s - C_{p2} - C_{p3})]}{C_s + C_{p3}/2}. \end{aligned} \tag{19}$$

Finally, the RHS of (19) is minimized when τ_1 is maximum. Since $\tau_1 \le t_1$, $t_2(C_s - C_{p2} - C_{p3}) + t_1(C_s - C_{p1}) \ge \frac{2F(C_s - C_{p2} - C_{p3}) + Kt_1}{C_s + C_{p3}/2}$, where $K = (C_s - C_{p1})(C_s + C_{p3}/2) - (C_{p1} + C_{p2})(C_s - C_{p2} - C_{p3})$. The sign of K depends on the strategy. We will show that choosing $K \le 0$ is sub-optimal.

Lemma 3. *For $N = 3$ and $M = 1$, then if $t_1 < t_2$ and $K > 0$ then, for f given by (8),*

$$\sum_{i=1}^{3} t_i \ge \frac{F}{C_s}\min\Big(f(C_2, C_3), f(C_3, C_2)\Big) \tag{20}$$

Lemma 4. *For $N = 3$ and $M = 1$, if $t_1 < t_2$ and $K \le 0$ then (2) holds again.*

The following lemma is proved using the sum-of-squares technique [12].

Lemma 5. *For $N = 3$ and $M = 1$, if (2) holds, then (20) holds with strict inequality.*