A Novel Grouping Strategy for Reducing Average Distribution Time in P2P File Sharing

Pui-Sze Tsang, Xiang Meng, King-Shan Lui
Department of Electrical and Electronic Engineering
The University of Hong Kong, Hong Kong
Email: {tsangps, h0880103, kslui}@eee.hku.hk

***Abstract*—Peer-to-Peer(P2P) file distribution has been widely used for file sharing in recent years. When compared with the traditional client-server model, the P2P model is a lot more efficient as each user can act as both a client and a server. This enables the P2P file distribution to scale well with increasing number of users. Grouping strategy has been introduced to reduce the *average distribution time* among peers without prolonging the total time needed to obtain a file. In this paper, a novel grouping strategy which groups peers of similar bandwidth together is introduced. We mathematically illustrate that under certain circumstances, this new grouping strategy performs better than the Greedy Grouping mechanism. To understand the performance of our grouping mechanism more comprehensively, we conduct extensive simulations. The results show that our mechanism can enhance the performance significantly in different network settings.**

I. Introduction

The rapid growth of Peer-to-Peer (P2P) networks in recent years has led to the development of many P2P applications.

The superiority of P2P model over client-server model can be demonstrated in terms of scalability and management. In a P2P model, any peer can download and upload file pieces from and to the others in the network. Peers may join or leave the network dynamically. Peers which have the whole file before the distribution are called *seeds*, while peers that have nothing initially and want to obtain the whole file are called *leechers*. A scheduling mechanism governs how the seeds and leechers share file pieces together. To measure how good a scheduling algorithm is, we study how long it takes for *all* leechers to obtain the whole file, that is the *file distribution time*. It is also the time requires for the last leecher to obtain the file. Another performance metric is the *average distribution time* which is used in [1]. It is the sum of finish times of all peers divided by the number of peers.

P2P file sharing model can be further classified into fluid model and chunk model. In the fluid model, peers can transmit a data bit to the others right after it has been received. Kumar and Ross derived the theoretical lower bound of the file distribution time on a stochastic fluid model [2]. They also developed an optimal file distribution scheme, referred as the KR-algorithm in this paper, which ensures all participating leechers complete at the same time t, and t is the theoretical lower bound of the distribution time. That is, there isn't any scheduling mechanism that completes the distribution before t. However, minimizing the finish time of the *last* leecher may not be optimal from *an individual* leecher's perspective. In the KR-algorithm, all leechers finish at the same time. If there is a leecher with very limited download capacity, it takes a long time for this slow leecher to complete the download process. Then, the KR-algorithm is not optimal from a fast leecher's perspective. To solve this problem, a grouping mechanism was proposed in [3], [4] in order to reduce the average distribution time without prolonging the total download time. Seeds and leechers are divided into groups. Only peers within the same group are allowed to share file pieces and inter-group communication is not permitted. The average distribution time is significantly reduced. However, the greedy approach makes a greedy choice in each step, which may not be optimal for the whole problem. For this reason, we propose a better grouping mechanism which puts leechers with similar download capacities into the same group. We show that this strategy can reduces the file distribution time more effectively.

The rest of this paper is organized as follows. Section II briefly introduces some related works. In Section III, a formal description of the problem is given. Mathematical analysis on the problem is provided in Section IV. Section V describes our leechers grouping algorithm. Simulation results are presented in Section VI. We conclude the paper in Section VII.

II. Related Work

File distribution time and *average distribution time* have been used by many studies as the performance metric. An analytic model is developed in [5] to study the performance of a few chunk-based tree architectures. A centralized optimal protocol is described in [6] in order to distribute a file consists of k pieces to n leechers within $k + log_2(n)$ time slots.

There have been a few analytical models based on the fluid model. Mundinger et al. derived a closed-form expression for the minimal time required to disseminate a file under a homogeneous environment where every peer has equal amount of uplink capacity and infinite downlink capacity [7]. This solution is far from general. Authors in [1] made use of the expression provided by Mundinger and designed an explicit file dissemination scheduling algorithm which minimizes the average finish time. However, their work also assumed that peers have infinite downlink capacities. Authors in [2] were the first to derive a general expression for the minimum achievable file distribution time in a heterogeneous fluid model. A scheduling algorithm was also designed to complete within the minimum distribution time. However, they assumed the system to be static, which means no peers join or leave during the distribution process. This is unrealistic and not beneficial for peers with abundant capacities, as they have to wait for other peers to finish. This motivates us to group the peers.

Grouping strategy has been proposed for web-cache and peers detection in [8], [9] and [10]. However, most of these mechanisms are based on the interest and geometrical information rather than bandwidth capacities of peers. The authors in [3], [4] proposed a grouping scheme based on the closed-form expression in [2]. Greedy algorithm is used to partition the peers. However, the greedy strategy only looks for a local optimal solution in each step. Besides, all the seeds available may not be fully utilized, leaving some idle seeds behind. In this paper, we propose a few general rules to improve the efficiency of grouping. A novel grouping algorithm based on these rules is developed.

III. Problem Description

A. Notation and Definition

In this section, we formally define our problem. We first define the notations used in the paper. We denote the set of seeds as S and the set of leechers as L. The upload bandwidth of a seed s, $s \in S$, and a leecher l, $l \in L$, are denoted as $u(s)$ and $u(l)$, respectively. Download bandwidth of l is denoted as $d(l)$. We further define $u(S) = \sum_{s\in S} u(s)$ and $u(L) = \sum_{l \in L} u(l)$ to be the sum of upload bandwidth of the seeds in S and leechers in L, respectively. $d^L_{min} = \min\{d(l) | l \in L\}$ is the minimum download bandwidth among all the leechers in set L. Leecher l simultaneously sends/receives to/from several peers, but the total transmission rate is either bounded by $u(l)$ or $d(l)$. We assume that the bottleneck lies on the senders and the receivers, but not the network in between them.

Kumar and Ross analyzed the minimum possible file distribution time of a static fluid model which peers do not join or leave during the process. We refer this time as the *minimum distribution time* for seeds in S to distribute file F to leechers in L, and denote it as $T_{min}(S, L, F)$, where:

$$T_{min}(S, L, F) = \max\{\frac{F}{d^L_{min}}, \frac{|L|F}{u(S) + u(L)}, \frac{F}{u(S)}\} \qquad (1)$$

$\frac{F}{d^L_{min}}$ is the time for the leecher with the minimum download bandwidth to obtain F. $\frac{|L|F}{u(S)+u(L)}$ reflects the time needed to distribute $|L|$ copies of F if both seeds and leechers contribute their upload bandwidth. $\frac{F}{u(S)}$ is the time required for seeds to send out a single copy of F. They also developed an optimal scheduling algorithm that allows the file distribution to end at the minimum distribution time. Using this KR-algorithm, all leechers complete at the same time. Let $T^A(l), l \in L$, be the finish time of any leecher l in set L using scheduling algorithm A. We define the *average distribution time* achieved by Algorithm A, $T^A_{avg}(S, L, F)$, to be $\frac{\sum_{l\in L} T^A(l)}{|L|}$. Let $T^{KR}(l)$ be the finish time of leecher l when the KR-algorithm is used. Then, $T^{KR}(l) = T_{min}(S, L, F)$ for every l and so $T^{KR}_{avg}(S, L, F)$ is also $T_{min}(S, L, F)$.

B. Grouping in fluid model

Consider the configuration in Table I. If F = 300M, by (1), $T_{min}(S, L, F) = T^{KR}_{avg}(S, L, F) = 1500$s. This means that every leecher in Table I requires 1500s to download the whole file using the KR-algorithm. To allow some leechers to finish early without prolonging the file distribution time, the authors in [3], [4] proposed the Greedy Grouping mechanism to divide the seeds and leechers into groups. Each group adopts the KR-algorithm and no inter-group communication is allowed. Each group may finish at a different time but should not exceed the time needed when grouping is not used, so faster leechers may finish earlier without waiting for the slow leechers.

The Greedy Grouping algorithm works as follows: seeds are sorted in ascending order of upload capacities, then they are placed into different groups such that the total seed upload bandwidth of each group is larger than d^L_{min}. Leechers are sorted according to the descending order of upload bandwidth, and then they are considered one by one. When leecher l_j is considered, it is assigned to the group that has the minimum file download time after including l_j. The Greedy Grouping algorithm divides the seeds and leechers of Table I into five groups where $G_1 = \{s_1, l_1, l_3, l_6\}$, $G_2 = \{s_2, l_2, l_4, l_5\}$, $G_3 = \{s_3\}$, $G_4 = \{s_4, l_9, l_{10}\}$ and $G_5 = \{s_5, l_7, l_8\}$. Let $T_{min}(G_z)$ be the minimum distribution time of group G_z, then the file distribution time of each group will be: $T_{min}(G_1) = 1500$s, $T_{min}(G_2) = 428.57$s, $T_{min}(G_3) = 0$s, $T_{min}(G_4) = 47.62$s, and $T_{min}(G_5) = 48.47$s. Although $T_{min}(S, L, F)$ is still 1500s, but the average distribution time using Greedy Grouping $T^G_{avg}(S, L, F) = (1500 \times 3 + 428.57 \times 3 + 0 + 47.62 \times 2 + 48.47 \times 2)/10 = 597.7883$s, which is less than $T^{KR}_{avg}(S, L, F) = 1500$s.

The example above shows that grouping looks promising in reducing the average distribution time. However, there are some situations where grouping cannot bring benefits. The work in [3], [4] show that grouping may be beneficial to a network configuration of seed set S and leecher set L only when $T_{min}(S, L, F) = \frac{F}{d^L_{min}}$. Otherwise, if $T_{min}(S, L, F) = \frac{|L|F}{u(S)+u(L)}$ or $\frac{F}{u(S)}$, then it is not beneficial to apply grouping.

TABLE I
Configuration of seeds and leechers

Seed	s_1	s_2	s_3	s_4	s_5
$u(s_i)$/Kbps	4100	5300	5700	10800	11300

Leecher	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8	l_9	l_{10}
$u(l_i)$/Kbps	600	500	1300	700	1100	1700	280	800	1500	300
$d(l_i)$/Kbps	200	700	1200	1400	2300	2700	6400	6700	6800	9800

C. Challenges on seeking an optimal solution

The authors in [3], [4] also proved that the problem of finding an optimal grouping that minimizes the average distribution time is NP-complete. One possible way to identify optimal grouping is by modeling the problem as an integer programming problem. Let $|S| = M$ and $|L| = N$. At most M groups can be formed. Let S_{xz} and L_{yz} be the variables indicating whether $s_x \in S$ and $l_y \in L$ are in group G_z, respectively. $S_{xz} = 1$ ($L_{yz} = 1$) means that s_x (l_y) is in group G_z; otherwise, $S_{xz} = 0$ ($L_{yz} = 0$). If there are k groups where $k \in [1, 2, ..., M]$, there are $k \times (M + N)$ number of variables.

We can then set up a binary integer program for every k, which aims at minimizing the average distribution time with $M+N$ constraints. The system of constraints is shown below.

$$\begin{array}{ll} \text{Min} & \frac{1}{N}\sum_{z=1}^{k}(T_{min}(G_z)|L_z|) \\ \text{Subject to} & \sum_{z=1}^{k} S_{xz} = 1 \ , \ x \in \{1, ..., M\} \\ & \sum_{z=1}^{k} L_{yz} = 1 \ , \ y \in \{1, ..., N\} \end{array}$$

The constraints imply that each seed and leecher can be assigned to at most one group. After the average distribution times of different k are found, the optimal grouping scheme can be identified. If the optimal k is 1, it means that grouping does not bring benefit at all. Due to the non-linear property of the objective function, the system is not easy to solve. This motivates us to look for other means to create groups.

IV. Analytical Results

In this section, we present two guidelines for developing a good grouping mechanism.

A. Groups should be smaller in size

To understand why more groups (group size becomes smaller) would reduce the average distribution time, we first study what the distribution time of a certain group is when grouping is applied. Given a seed set S and a leecher set L, if $T_{min}(S, L, F) = \frac{F}{d^L_{min}}$, the authors in [3], [4] mathematically illustrated that grouping may be beneficial. Therefore, it is intuitive that we should divide a configuration into smaller groups until each smaller group (S, L) can no longer be divided to improve the performance. That is, $T_{min}(S, L, F) \neq \frac{F}{d^L_{min}}$ or S contains one seed only. If S contains more than one seed, it is very likely that $T_{min}(S, L, F) = \frac{F}{d^L_{min}}$. Because $u(S)$ is the sum of the capacities of all the seeds and $\frac{u(L)}{|L|-1}$ is more or less the average uplink capacity of a leecher, it is highly likely that $u(S) \geq \frac{u(L)}{|L|-1}$. Then,

$$T_{min}(S, L, F) = \frac{|L|F}{u(S) + u(L)} = \frac{F}{\frac{u(S)+u(L)}{|L|}} \tag{2}$$

Fig. 1 illustrates the change in $\frac{u(S)+u(L)}{|L|}$ with increasing $|L|$ for different ranges of leecher upload capacities. The range is $[20Kbps, (20 \times q)Kbps]$ and q varies from 5 to 180, while the number of seeds is fixed at 5. From the figure we observed that $T_{min}(S, L, F)$ in (2) decreases as $|L|$ decreases. It implies that having smaller groups ensures that each group has fewer leechers, and hence $T_{min}(S, L, F)$ can be reduced.

B. Group leechers with similar capacities together

We now study which leechers should be put in the same group to further reduce the file download time. Consider a case where there are two groups: G_x and G_y. G_x consists of seed set S_x and leecher set L_x, while G_y consists of seed set S_y and leecher set L_y. There are two new incoming leechers:

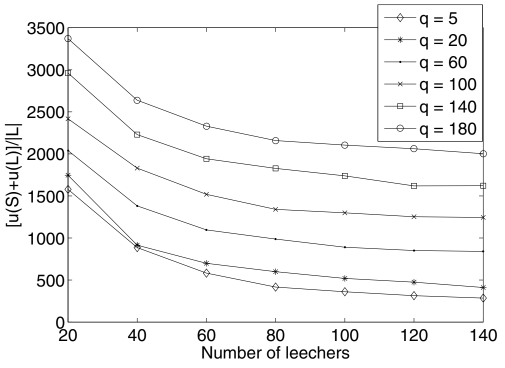


Fig. 1. Relation between $|L|$ and $\frac{u(S)+u(L)}{|L|}$ when $|S| = 5$.

l_a and l_b. Let $d^{L_x}_{min} < d(l_a) < d(l_b) < d^{L_y}_{min}$. Suppose that we want to put one leecher in G_x and one in G_y. We want to know whether l_a should be put in G_x or l_a should be put in G_y. Our goal is to compare the difference in these two assignments: [A] l_a is assigned into L_x and l_b is assigned into L_y, in [B] l_b is assigned into L_x and l_a is assigned into L_y.

In Assignment [A], we have,

$$T_{min}(S_x, L_x \cup \{l_a\}, F) = \frac{F}{\min\{d^{L_x}_{min}, \frac{u(S_x)+u(L_x)+u(l_a)}{|L_x|+1}, u(S_x)\}} \tag{3}$$

$$T_{min}(S_y, L_y \cup \{l_b\}, F) = \frac{F}{\min\{d(l_b), \frac{u(S_y)+u(L_y)+u(l_b)}{|L_y|+1}, u(S_y)\}} \tag{4}$$

In Assignment [B], we have,

$$T_{min}(S_x, L_x \cup \{l_b\}, F) = \frac{F}{\min\{d^{L_x}_{min}, \frac{u(S_x)+u(L_x)+u(l_b)}{|L_x|+1}, u(S_x)\}} \tag{5}$$

$$T_{min}(S_y, L_y \cup \{l_a\}, F) = \frac{F}{\min\{d(l_a), \frac{u(S_y)+u(L_y)+u(l_a)}{|L_y|+1}, u(S_y)\}} \tag{6}$$

We compare the two assignments using two cases. In the first case, we assume that $u(l_a) \leq u(l_b)$. In the second case, we assume that $u(l_a) > u(l_b)$. In each case, we compare the minimum distribution times of G_x when l_a and l_b are assigned to L_x independently. That is, we compare (3) and (5). Similarly, we compare (4) and (6). We now describe how to compare (3) and (5) in details in the first case: $u(l_a) \leq u(l_b)$. Since there are three terms in formulas (3) and (5), so there are 3×3 combinations in comparing (3) and (5) as shown in Table II. Due to space limitation, we only explain some representative situations: those entries marked with (i), (ii), (iii) and (iv), in details. For example, (i) in Table II represents the situation where both (3) and (5) equal to $\frac{F}{d^{L_x}_{min}}$.

Each entry tells whether putting l_a or l_b into the group results in a smaller distribution time. l_a means that putting l_a in the group is better. Similarly, l_b means that assigning l_b in the group has a smaller distribution time. S means that the distribution times of assigning l_a and l_b are the same. Some entries are marked as *nil*, meaning such scenario never happens. When we are not certain, we use U to represent it.

Let's consider Situation (i) in Table II. Since both (3) and (5) equal to $\frac{F}{d^{L_x}_{min}}$, the minimum distribution times of

(3) and (5) are the same. Now, we consider Situation (ii). (3) = $\frac{F}{d^{L_x}_{min}} \geq \max\{\frac{(|L_x|+1)F}{u(S_x)+u(L_x)+u(l_a)}, \frac{F}{u(S_x)}\}$. If l_b is assigned into L_x instead, given that $u(l_b) \geq u(l_a)$, we have $\frac{(|L_x|+1)F}{u(S_x)+u(L_x)+u(l_a)} \geq \frac{(|L_x|+1)F}{u(S_x)+u(L_x)+u(l_b)}$. Therefore, we can ensure that (5) = $\frac{F}{d^{L_x}_{min}} \geq \max\{\frac{(|L_x|+1)F}{u(S_x)+u(L_x)+u(l_b)}, \frac{F}{u(S_x)}\}$, which means that $\frac{(|L_x|+1)F}{u(S_x)+u(L_x)+u(l_b)} > \frac{F}{d^{L_x}_{min}}$ never happens.

For (iii), (3) = $\frac{(|L_x|+1)F}{u(S_x)+u(L_x)+u(l_a)} \geq \max\{\frac{F}{d^{L_x}_{min}}, \frac{F}{u(S_x)}\}$. Since (5) = $\frac{F}{d^{L_x}_{min}}$, we can ensure that (5) $\leq$ (3). It means that assigning l_b to L_x must not be worse than assigning l_a to L_x. So, we put l_b in this entry. In (iv), we compare (4) and (6). As it is given that $d(l_a) < d(l_b)$, (6) = $\frac{F}{d(l_a)} > \frac{F}{d(l_b)}$ = (4). Hence, it is better to assign l_b to L_y for (iv).

TABLE II
PERFORMANCE OF ASSIGNMENT [A] AND [B] FOR $u(l_a) \leq u(l_b)$

For G_x,

		(5)		
		$\frac{F}{d^{L_x}_{min}}$	$\frac{(\|L_x\|+1)F}{u(S_x)+u(L_x)+u(l_b)}$	$\frac{F}{u(S_x)}$
(3)	$\frac{F}{d^{L_x}_{min}}$	S $^{(i)}$	nil $^{(ii)}$	nil
	$\frac{(\|L_x\|+1)F}{u(S_x)+u(L_x)+u(l_a)}$	l_b $^{(iii)}$	l_b	l_b
	$\frac{F}{u(S_x)}$	nil	nil	S

For G_y,

		(6)		
		$\frac{F}{d(l_a)}$	$\frac{(\|L_y\|+1)F}{u(S_y)+u(L_y)+u(l_a)}$	$\frac{F}{u(S_y)}$
(4)	$\frac{F}{d(l_b)}$	l_b $^{(iv)}$	l_b	nil
	$\frac{(\|L_y\|+1)F}{u(S_y)+u(L_y)+u(l_b)}$	l_b	l_b	nil
	$\frac{F}{u(S_y)}$	l_b	l_b	S

TABLE III
PERFORMANCE OF ASSIGNMENT [A] AND [B] FOR $u(l_a) > u(l_b)$

For G_x,

		(5)		
		$\frac{F}{d^{L_x}_{min}}$	$\frac{(\|L_x\|+1)F}{u(S_x)+u(L_x)+u(l_b)}$	$\frac{F}{u(S_x)}$
(3)	$\frac{F}{d^{L_x}_{min}}$	S	l_a	nil
	$\frac{(\|L_x\|+1)F}{u(S_x)+u(L_x)+u(l_a)}$	nil	l_a	nil
	$\frac{F}{u(S_x)}$	nil	l_a	S

For G_y,

		(6)		
		$\frac{F}{d(l_a)}$	$\frac{(\|L_y\|+1)F}{u(S_y)+u(L_y)+u(l_a)}$	$\frac{F}{u(S_y)}$
(4)	$\frac{F}{d(l_b)}$	l_b	nil	nil
	$\frac{(\|L_y\|+1)F}{u(S_y)+u(L_y)+u(l_b)}$	U	l_a	l_a
	$\frac{F}{u(S_y)}$	l_b	nil	S

In Table II, we see that both G_x and G_y favor l_b in some situations but none of them favor l_a. It is reasonable and intuitive since l_b has larger uplink and downlink capacities than l_a. However, we cannot assign l_b to both L_x and L_y. Therefore, we consider in which group l_b is more likely to bring benefit and assign l_b to this group, while assigning l_a to the other one. Since the frequency that l_b appears in the table of G_y is higher than the table of G_x in Table II, it is more beneficial to assign l_b to G_y and l_a to G_x. On the other hand, in Table III, G_x favors l_a and G_y works well with both l_a and l_b. In that case, we should assign l_a to G_x and l_b to G_y.

In conclusion, clustering leechers with similar downlink capacities together leads to a better result in most cases.

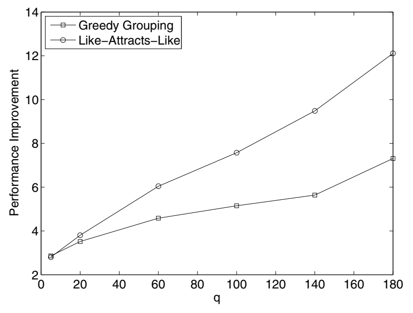
V. LIKE-ATTRACTS-LIKE PROTOCOL

Our grouping mechanism works in two phases: seed grouping and leecher grouping. We adopt the same seed grouping mechanism described in [3], [4]. Initially, each group contains only one seed. The groups are sorted in ascending order of the total upload bandwidth of the seeds in the group. If $u(S_i)$ in group G_i is less than d^L_{min}, we merge G_i and G_{i+1} to form G'_i. Then, we sort all groups in an ascending order of their upload bandwidth again. This process terminates when the total upload bandwidth of each group is no less than d^L_{min}.

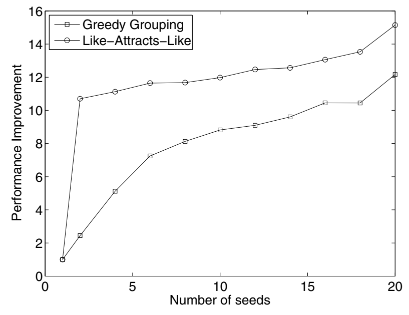
Assume k groups have been derived from the seed grouping process. We now assign the N leechers in set L to these k groups. Our Like-Attracts-Like algorithm, which is shown in Algorithm 1, is designed according to the general rules listed in Section IV. We first sort the leechers according to download capacities in ascending order. We label the leechers as l_1, ... , l_N where $d(l_i) \leq d(l_j)$ if $i \leq j$. We also sort the seed groups in an ascending order of their capacities. We want to partition the leechers into k groups, one for each seed group, according to the *group leechers with similar capacities together* strategy. That is, if l_i is in seed group G_x, l_j cannot be in G_y if $j \leq i$ and $y > x$. For example, if l_4 is assigned to G_5, then l_5 cannot be assigned to G_1, G_2, G_3 and G_4. Due to this property, when we want to partition leechers into k groups, we need to find $k-1$ *partition indices* $I_1, I_2, ..., I_{k-1}$ such that $l_1, ..., l_{I_1}$ are assigned to G_1, $l_{I_1+1}, ..., l_{I_2}$ are assigned to G_2, and so on.

Assume that the leechers in groups G_{i-1} and G_i are $l_m, l_{m+1}, ..., l_{n-1}, l_n$. The partition index I_{i-1} must fall in the range $[m, n-1]$. We determine I_{i-1} by finding which number in $[m, n-1]$ results in the minimum $T_{sum} = |L_{i-1}|T_{min}(S_{i-1}, L_{i-1}, F) + |L_i|T_{min}(S_i, L_i, F)$. We find the partition indices in descending order. That is, we first find I_{k-1} and determine the partition between L_{k-1} and L_k. In order to ensure that $G_1, G_2, ..., G_{k-2}$ have at least one leecher in each of them, we can at most assign $l_{k-1}, l_k, l_{k+1}, ..., l_N$ to L_{k-1} and L_k. Therefore, we find I_{k-1} from the range $[k-1, N-1]$ based on T_{sum}. After finding I_{k-1}, we proceed to find I_{k-2} which determine the partition between L_{k-2} and L_{k-1}. Again, we need to ensure there is at least one leecher in each of the groups G_1, ..., G_{k-3}. Therefore, I_{k-2} must be in the range $[k-2, I_{k-1}-1]$. We repeat the procedure until all the partition indices are found. As k groups are formed after seed grouping, the time complexity of Like-Attracts-Like is $O(NlogN) + O(kN)$ if heap sort is used.

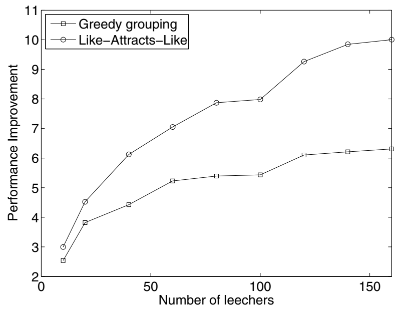
Consider Table I, let F = 300M and 5 groups are derived from seed grouping: $G_1 = \{s_1\}$, $G_2 = \{s_2\}$, $G_3 = \{s_3\}$, $G_4 = \{s_4\}$ and $G_5 = \{s_5\}$. We first find I_4, which falls in $[4, 9]$ as there are 10 leechers. Table IV shows the value of T_{sum} when I_4 equals to 4, 5, ..., 9. We observed that T_{sum} is the smallest when $I_4 = 6$, so we assign l_7, l_8, l_9, l_{10} to G_5. We then move on to I_3, which ranges from 3 to $I_4 - 1 = 5$. This process terminates when we reach G_1. Finally, we have: $G_1 = \{s_1, l_1\}$, $G_2 = \{s_2, l_2\}$, $G_3 = \{s_3, l_3, l_4\}$, $G_4 = \{s_4, l_5, l_6\}$, and $G_5 = \{s_5, l_7, l_8, l_9, l_{10}\}$. The average distribution time using Like-Attracts-Like is $T^{LL}_{avg}(S, L, F)$ = 302.7946s, which



(a) Effect of q ratio



(b) Effect of number of seeds M



(c) Effect of number of leechers N

Fig. 2. Simulation Results of grouping in Fluid model

is smaller than $T^G_{avg}(S, L, F) = 597.7883$s.

TABLE IV
T_{sum} OVER VARYING L_5 AND L_4

L_4	$\{l_4\}$	$\{l_4, l_5\}$	$\{l_4, ..., l_6\}$	$\{l_4, ..., l_7\}$	$\{l_4, ..., l_8\}$	$\{l_4, ..., l_9\}$
L_5	$\{l_5, ..., l_{10}\}$	$\{l_6, ..., l_{10}\}$	$\{l_7, ..., l_{10}\}$	$\{l_8, ..., l_{10}\}$	$\{l_9, l_{10}\}$	$\{l_{10}\}$
T_{sum}	996.89	984.13	981.36	1051.39	1163.03	1316.33

To evaluate an algorithm, say Algorithm A, we use the performance improvement ratio ρ defined in (7).

$$\rho = \frac{T_{min}(S, L, F)}{T^A_{avg}(S, L, F)} \tag{7}$$

As $T^{KR}_{avg}(S, L, F) = T_{min}(S, L, F)$ if grouping is not applied, so the ρ of the KR-Algorithm is 1. Using Greedy Grouping, $\rho = \frac{1500}{597.8} = 2.5092$ for the network in Table I. For Like-Attracts-Like, $\rho = \frac{1500}{302.8} = 4.9539$, which is a lot better.

Algorithm 1 Like-Attracts-Like (Leecher Grouping Part)

```
Input: (1) $k$ seed groups $S_1, S_2, ..., S_k$, where $k > 1$;
       (2) download bandwidth of $N$ leechers, where $d(l_1) < d(l_2) < ... < d(l_N)$;
Output: Partition indices $I_1, I_2, ..., I_{k-1}$;
 1: $I_1 = I_2 = ... = I_{k-1} = I_{Min} = N - 1$;
 2: for g = $k - 1$ downto 1 do
 3:     $Min = \infty$;
 4:     for i = $g$ to $I_g$ do
 5:         Leecher set $L_{g+1} = \{l_{i+1}, ..., l_{I_g}, l_{(I_g)+1}\}$;
 6:         Leecher set $L_g = \{l_g, ..., l_i\}$;
 7:         $T_{sum} = |L_g|T_{min}(S_g, L_g, F) + |L_{g+1}|T_{min}(S_{g+1}, L_{g+1}, F)$;
 8:         if $Min > T_{sum}$ then
 9:             $Min \leftarrow T_{sum}$;
10:             $I_{Min} \leftarrow i$;
11:         end if
12:     end for
13:     $I_1 = I_2 = ... = I_g \leftarrow (I_{Min} - 1)$;
14: end for
```

VI. Simulation Results

We conduct simulations on a wide range of configurations. We vary the capacities of peers, plus the numbers of seeds and leechers. Number of seeds varies over $[1, 20]$ while number of leechers falls into $[10, 180]$. Upload capacities of seeds vary over $[56Kbps, 12Mbps]$. 56Kbps is the speed of dial-up service while 12Mbps is the speed of ADSL2 [11]. We set the upload bandwidth of leechers to vary over $[p, p \times q]$ where $p = 20Kbps$ and q varies over $[5, 180]$. Each data point in the graphs is the average result of 30 random configurations.

In Fig. 2(a), we vary the upper bound of the uplink capacity of leechers (q). The configurations contain 5 seeds and 100 leechers. In Fig. 2(b), we vary the number of seeds (M). Both q and the number of leechers (N) are 100. Lastly, Fig. 2(c) studies the variation in the number of leechers on the performance. q is fixed at 100 and the number of seeds is 5.

We observe that the performance of the Like-Attracts-Like is always better than the Greedy Grouping. The deviation in performance increases rapidly as q, M and N increase.

VII. Conclusion

Optimization of the average distribution time is complicated, and Greedy Grouping cannot guarantee that a global optimal solution can be found. Hence, we proposed two guidelines to improve the efficiency of grouping. A new leechers grouping algorithm, Like-Attracts-Like, is designed according to the guidelines mentioned. We also demonstrated that Like-Attracts-Like often performs better than Greedy Grouping.

References

[1] G. M. Ezovski, Ao Tang, and Lachlan L.H. Andrew, "Minimizing average finish time in p2p networks," in *Proc. of the INFOCOM 2009*, pp. 594–602.
[2] R. Kumar and K. Ross, "Peer assisted file distribution: The minimum distribution time," in *Proc. of the IEEE HOTWEB 2006*, pp. 1–11.
[3] Lingjun Ma, Xiaolei Wang, and King-Shan Lui, "A novel peer grouping scheme for p2p file distribution networks," in *Proc. of the ICC 2008*, pp. 5598–5602.
[4] Lingjun Ma, Pui-Sze Tsang, and King-Shan Lui, "Improving file distribution performance by grouping in peer-to-peer networks," in *IEEE Transactions on Network and Service Management*, September 2009, vol. 6, Issue 3, pp. 149–162.
[5] E. W. Biersack, P. Rodriguez, and P. Felber, "Performance analysis of peer-to-peer networks for file distribution," in *Proc. of the 5th International Workshop on QofIS 2004*, 2004, pp. 1–10.
[6] S. Sanghavi, B. Hajek, and L. Massoulie, "Gossiping with multiple messages," in *Proc. of the INFOCOM 2007*, pp. 2135–2143.
[7] G. Mundinger, R. Webb, and G. Weiss, "Analysis of peer-to-peer file dissemination," in *Proc. of the 8th Workshop on MAMA 2006*, pp. 12–14.
[8] K Kojima, "Grouped peer-to-peer networks and self-organization algorithm," in *Proc. of the IEEE SMC 2003*, pp. 2970–2976.
[9] Xiaole Bai, Shuping Liu, Peng Zhang, and R. Kantola, "ICN: interest-based clustering network," in *Proc. of the 4th International Conference on Peer-to-Peer Computing*, 2004, pp. 219–226.
[10] Yun He, Jianzhong Zhang, Xiaoguang Niu, and Qi Zhao, "Search and index in locality-based clustering overlay," in *Proc. of the IEEE CCGrid 2005*, pp. 229–236.
[11] Official Web Site of International Telecommunication Union, "http://www.itu.int/rec/t-rec-g.992.3/en," 2009.