

Distributed Pairing for File Sharing in Large-Scale Peer-to-Peer Networks

Chih-Lin Hu and Da-You Chen

Department of Communication Engineering, National Central University, Taoyuan, Taiwan 32001, R.O.C.
clhu@ce.ncu.edu.tw; 995203011@cc.ncu.edu.tw

Abstract—Peer-to-peer systems encounter critical issues, such as peer churn, free riding, and varied workload, which significantly affect performance results such as resource utilization, service agility, and fairness. This study proposes a distributed pairing scheme that supports fast and fair file sharing applications in large-scale peer-to-peer networks. This design exploits the peer duality of both server-oriented peer capacity and client-oriented peer contribution. Accordingly, the scheme can perform peer assignment, substitution, and elimination, and cope with subtle situations of free-riding, service starvation, and download blocking, thus making the system design robust and amenable. Simulation results confirm that the proposed mechanisms are effective in maintaining service agility and fairness in large-scale, distributed peer-to-peer file sharing environments.

I. INTRODUCTION

Peer-to-peer (P2P) networking has emerged as a new distributed computing paradigm for content distribution applications. The wide deployment of P2P systems has an explosive increase of peer population and service capacity. In P2P file sharing applications, like Napster, Gnutella, eDonkey, and BitTorrent, a peer requests files from its peers, and also stores and serves files to its peers [1]. Increasing peer population not only increases request workload, but also produces a concomitant increase in service capacity to process the workload [2].

However, critical issues such as peer churn, free riding, and varied workload significantly affect performance results such as service agility, fairness, and resource utilization. In P2P systems, peers cooperate to run an application-level overlay network [3] that provides connectivity, signal messaging, routing, and searching between end hosts that are addressable on top of IP communication networks. *Peer churn*, which represents the dynamics of peer participation in a system, influences overlay design, resiliency, and assessment [4]. Peer transiency and its mutual dependency affect the efficacy of replication, search, and query mechanisms in content distribution [5][6]. On the other hand, *free riders*, which maximize their own utility disproportionate to their contribution to the system, negate the reciprocal provision of resources among peers [7]. If no request admission or scheduling policy is enforced to maintain service fairness, generous peers often become frustrated and refuse to contribute anymore [8]. To encourage peer contribution and guarantee service fairness, the growing recognition of reputation-based incentive schemes [9] has generated significant research in this area: reputable peers receive high priority and are rewarded more resources to compensate for their contribution.

This study proposes a distributed pairing strategy for fast and fair file sharing in large-scale P2P networks. This design exploits *peer duality*, in that each peer plays a dual role of both service provider (server) and a consumer (client) of the implemented service [10] in P2P networks. Accordingly, the system performs distributed peer management that records peers' uploading and downloading behavior. The system then applies distributed pairing schemes based on reputation incentives to control the request admission and scheduling processes. The proposed mechanisms guarantee fair and efficient use of resources without decreasing overall system throughput against peer churn, free riding, and workload dynamics.

Therefore, the concept, design, and development phases are described as follows. First, a practical large-scale P2P system usually consists of many small-scale centralized and structured P2P systems [1][11]. Each small system contains a hosting tracker that acts as a local central directory for peers in its domain. Trackers from external domains form in a tracker network and can forward downloading requests to other trackers. Second, with the grading measures of client-oriented peer contribution and server-oriented peer capacity, the trackers can manage peers in a distributed manner to cope with free-riding, service starvation, and download blocking. Third, this study presents two kinds of tracker-oriented and peer-oriented pairing schemes based on indirect-reciprocity incentives [12] to control the request admission and scheduling processes for fair and efficient use of resources. Finally, experimental simulations assess performance sensitivities in terms of average download time, download ratio by contribution range, and communication overhead. Consequently, this study presents a systematic methodology that provides an effective peer pairing scheme in large-scale distributed P2P file sharing systems. The proposed mechanisms protect system throughput, service agility, and fairness against service starvation and varying traffic dynamics in P2P networks.

The rest of this article is organized as follows. Section II models a distributed P2P environment. Section III details the proposed distributed pairing schemes. Section IV presents the performance results, while Section V provides the conclusion.

II. SYSTEM MODEL

This section models a large-scale distributed P2P network, and the measures of client-oriented contribution and server-oriented bandwidth capability for priority determination and pairing assignment among peers.

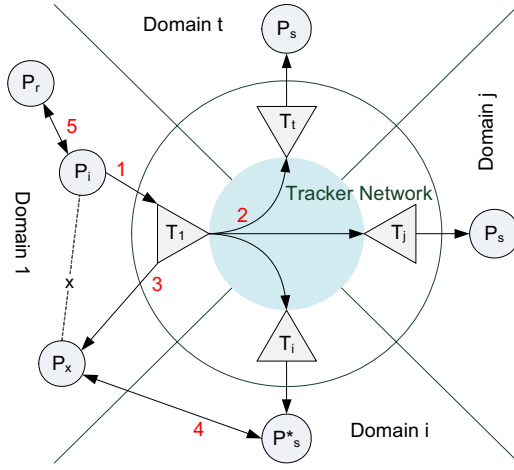


Fig. 1. Distributed pairing strategies in large-scale P2P networks.

A. Two-tier System Architecture

The practical deployment of large-scale P2P systems is based on the partially-centralized and structured P2P model [1]. As Fig. 1 shows, this reference architecture can be seen as an instance of a two-tier system that consists of many small-scale centralized and structured P2P systems. Every small P2P system contains a hosting tracker T_i in charge of peers that exist in its domain D_i . Trackers from external P2P domains form and communicate in a tracker group TG , called a “closed tracker network.” When a peer p_i in D_i requests a segment S_k , $T_i(S_k)$ represents the hosting tracker that handles this request. Each tracker can forward queries and downloading requests to other trackers in the tracker network. Since a tracker network is relatively small, this design plainly refers to a simple request/response messaging protocol by unicasting and broadcasting messages over the tracker network. This type of network appears in many BitTorrent-like P2P applications.

In a tracker network each tracker acts as a local central repository, and performs segment management, peer management, and service provision for peers in its local domain [11]. For every S_k in D_i , $T_i(S_k)$ records a set of peers that own this segment, and manages these peers with their states in service. Thus, every peer in D_i connects to $T_i(S_k)$ and inquires about where any indicated segment can be downloaded. $T_i(S_k)$ communicates with other trackers to collaborate on distributed pairing over the tracker network. Then, $T_i(S_k)$ selects one out of many candidate server peers in different domains. With a request reply, a client peer directly connects to its assigned server peer to access the indicated segment.

B. Formulation: Peer's State, Capability, and Grading

In P2P systems the design of resource utilization is based on a slotted time model to allocate the upload and download bandwidth. Each time slot has a minimal transfer rate μ as a base unit of bandwidth allocation. Accordingly, each file is divided into a number of equally-sized segments; it takes one time slot to deliver each file segment. For simplicity, let a peer

p_i have uniform upload and download bandwidth capacities with a maximal data transfer rate μ_i . In this case, p_i can at most offer $\lfloor \mu_i/\mu \rfloor$ transfer sessions simultaneously in either download or upload way. Peers that have a remaining transfer rate of less than μ will not be allowed to request or offer any segments until they have sufficient bandwidth.

Selecting a fit server peer is a decisive step before a client peer starts to download any segment. The system performs peer management according to upload and download states of each peer, as Definitions 1 and 2 specify. During a download/upload session between a pair of client and server peers, $T_i(S_k)$ monitors and adjusts bandwidth allocation in a timely manner. Meanwhile, $T_i(S_k)$ records the download/upload state of every requested segment, remaining upload bandwidth, and uploading contribution in its repository.

Definition 1 (upload state) A server peer falls into one of three states: busy, normal, and leisure. A busy peer is unable to supply any more segments because its remaining upload bandwidth is less than μ . A normal peer is uploading segments to other peers and has a remaining upload bandwidth higher than μ . A leisure peer has an upload bandwidth higher than μ , but is not currently uploading any segments.

Definition 2 (download state) A client peer falls into one of three states: active, sleep, and idle. A download state is active when a peer requests a segment, successfully finds it, and negotiates with a server peer to process the downloading. If a server peer is found, but unavailable temporarily, the state is set to sleep. Otherwise, a peer is idle if no requests are pending or no server peer is found.

The system adopts an indirect-reciprocity-based incentive approach to encourage peers to contribute their own resources to achieve better performance. This approach refers to the historical and accumulated contributions of upload bandwidth voluntarily provided by the peer. Without loss of generality, the measure of peer contribution is given by a linear formula of a peer's previous contribution:

$$C_i = UL_{past}(i) \times \alpha + UL_{now}(i) \times (1 - \alpha), \quad (1)$$

where UL_{now} is the number of segments which a peer is uploading now, UL_{past} is the number of segments that a peer has uploaded in the past, and $0 \leq \alpha \leq 1$ is a tunable parameter of relative weighting between the terms UL_{past} or UL_{now} that can be emphasized as desired. Weighting UL_{past} favors a peer that stays in the system for a long time. Comparatively, weighting UL_{now} favors a peer that has superior upload capacity, popular segments, or contains more segments in its local storage. The central server takes (1) to prioritize download requests from client peers.

On the other hand, in allocating upload bandwidth, a central server decides which server peer should serve client peer's requests by grading the upload bandwidth capacity that a server peer can provide to its client peers, given by

$$G_i = \mu_i / (UL_{now}(i) + 1), \quad \text{and } G_i \geq \mu. \quad (2)$$

III. DISTRIBUTED PAIRING SCHEME

This section describes a distributed pairing scheme for large-scale, distributed P2P file sharing systems. This design presents two kinds of tracker-oriented and peer-oriented distributed pairing approaches that are functional in maintaining service agility and fairness in dynamic network environments.

A. Design Considerations

The practical deployment of a large-scale distributed P2P system comprises many small-scale centralized and structured P2P systems. Though an unstructured P2P system is somewhat workable in a small-scale network, combining many unstructured P2P systems into a large one is impracticable due the drastic increase of message workload and tremendous search requests flooding the system. Instead, the proposed scheme is based on a partially-centralized and structured network architecture. Figure 1 shows this reference architecture, which introduces a new tracker network. A tracker group includes many trackers that perform peer assignment strategies in their local P2P domains. Trackers communicate with other trackers to collaborate on the distributed peer pairing scheme over the tracker network. Due to a variety of architectural and protocol dependencies, the design of a tracker network overlay is beyond the scope of this study. Since a tracker network is relatively small, without complication, this design plainly refers to a simple request/response messaging protocol by unicasting and broadcasting.

The proposed distributed peer pairing scheme augments the centralized peer assignment strategies in our previous study [11], which are particularly basic peer selection scheme (BPSS) and advanced peer selection strategies with peer substitution and elimination (APSS and APSS-E). The below summarizes the effects of BPSS, APSS, and APSS-E, while the details are omitted here to save page space.

- BPSS chooses server peers to process download requests in descending order of peer contribution in light of indirect-reciprocity notion [12]
- APSS extends BPSS with a substitute policy to cope with peer churn and varying request workload.
- APAS-E associates APSS with a peer elimination process to cope with possible service starvation and further boost the ability of service differentiation and fairness.

The distributed pairing scheme is applied into local and external P2P domains, simultaneously and collectively, to deal with service fairness, throughput, and starvation in a distributed manner. Note that the BPSS procedure within a local domain remains unchanged. What is mainly modified in the proposed scheme is the procedure of generating a set of candidate server peers from a tracker group instead of a singular tracker, as compared in APSS. The proposed scheme is based on several assumptions that are appropriate in a large-scale, distributed P2P environment.

- A peer is able to communicate with more than one tracker to operate different queries and downloading requests.

- When a peer has completely downloaded any requested segment through some tracker's coordination, this peer becomes linked to the domain of that tracker. In other words, a peer can concurrently belong to multiple domains, hosted by different trackers, from which this peer has downloaded different segments.
- Every tracker is not only responsible for offering the segments kept its local domain, but can also deliver to and receive from other trackers specific request/response messages to perform distributed tasks.

B. Scheme Description

The proposed distributed pairing scheme extends BPSS, APSS, and APSS-E with elegant modification for fast, fair and non-blocking file sharing in distributed P2P environments. This scheme performs two-phase procedures, BPSS and distributed version of APSS and APSS-E, in local and global P2P domains, respectively.

1) *Local Domain*: Each tracker T_i in its local domain D_i takes the request workload in descending order of peer grading G_i , and schedules requests in descending order of peer contribution C_i . Given a peer p_i whose download bandwidth is higher than μ , it can send T_i a request r_k^i to download a segment S_k . With a r_k^i , T_i generates a peer set, called $H(S_k)$ including all peers having S_k^i in their local storages. Then, T_i selects the server peer p_r with the highest upload rate in $H(S_k)$. If p_r 's G_r is not lower than μ , T_i notifies p_i of its server peer p_r in charge. Then, p_i negotiates with p_r for downloading S_k . If p_r has no enough upload bandwidth to serve r_k^i at this moment, T_i will mark p_i 's download state as sleep and push this request back into the request queue. The server later checks if r_k^i is valid yet. In this way, T_i can simply and evenly balance the use of bandwidth capacities among peers in D_i .

However, the basic scheme can encounter upload or download blocking due to service starvation, resulting in poor performance. This situation does not mean that a requested segment does not exist in the system. A heavy request workload for scarce segments, peer churn, and skewed access patterns can all induce this problem. When all peers owning S_k are in busy state, they have no more bandwidth to upload. Any requests for S_k are then blocked until any p_r in $H(S_k)$ reclaims sufficient upload bandwidth to accept a new request.

2) *Global Domain*: The proposed distributed pairing scheme enforces the peer substitute and elimination policies to address this problem of service starvation. This scheme attempts to find an optimal substitute peer p_s^* in the distributed system to replace a busy peer p_r in $H(S_k)$ and take over its ongoing uploading tasks. The replaced peer p_r can then reclaim more upload bandwidth and accept another request for some segment now held by p_r .

In reference to Fig. 1, the distributed pairing scheme has five procedural steps (Steps 1–5). Algorithm DP shows the algorithmic form to ease presentation. Initially, distributed pairing will start in case that a peer p_i with a request r_k^i runs into a downloading blocking situation.

Step 1: The hosting tracker $T_i(S_k)$ first determines $H(S_k)$, including peers in D_i that own S_k , and then generates $B(S_k)$, i.e., a super set of segments which all peers in $H(S_k)$ are currently delivering.

Step 2: With $B(S_k)$, $T_i(S_k)$ attempts to determine an optimal substitute server peer p_s^* in the distributed P2P system. In doing so, $T_i(S_k)$ not only finds a local substitute server peer p_s^i in D_i , but also asks other trackers to find and report external substitute server peers, respectively. Thus, $T_i(S_k)$ gathers a set of all potential substitutes $H^*(TG)$ from all trackers in TG . The peer with the highest grade in $H^*(TG)$ is chosen as p_s^* .

To produce p_s^* , this study presents the *tracker-oriented* and the *peer-oriented* methods, respectively described as follows.

a) *Tracker-oriented method:* The hosting tracker $T_i(S_k)$ appeals to all external trackers in the tracker network to collaboratively find p_s^* . Below is the procedure of this method.

- 1) $T_i(S_k)$ broadcasts $H(S_k)$ to other trackers in the tracker network. Every external tracker T_x checks if there are any peers in $H(S_k)$ which also exist in its domain D_x . If no peer is found, T_x sends a negative report to $T_i(S_k)$. If some peers in $H(S_k)$, denoted as $H_{sub}(S_k)$, are found in D_x , T_x assembles a subset of $B(S_k)$, denoted as $B_{sub}(S_k)$, which peers in $H_{sub}(S_k)$ are concurrently delivering.
- 2) T_x tries to find a candidate server peer for every segment in $B_{sub}(S_k)$. T_x then has a set of candidate server peers, $H^*(B_{sub}(S_k))$. The peer with the highest grade is regarded as the potential substitute p_s^x in D_x . Thus, T_x reports p_s^x to $T_i(S_k)$.
- 3) Since all external trackers perform the above process simultaneously, $T_i(S_k)$ can receive a number of reports from external trackers. Accordingly, $T_i(S_k)$ figures out the optimal substitute p_s^* .

b) *Peer-oriented method:* The hosting tracker $T_i(S_k)$ requests local peers in $H(S_k)$ to report a list of potential trackers which $T_i(S_k)$ will later ask to collaboratively find p_s^* . The procedure of this method is given below.

- 1) In light of $H(S_k)$ and $B(S_k)$, $T_i(S_k)$ asks each peer in $H(S_k)$ to report a list of pairing records $\langle \text{segment}, \text{tracker} \rangle = \langle S_x, T_x \rangle$. A pairing record means that a peer in $H(S_k)$ is delivering a segment S_x in $B(S_k)$ under a tracker T_x 's coordination. Thus, $T_i(S_k)$ learns a subset of potential trackers TG_{sub} that are responsible for delivering $B(S_k)$.
- 2) $T_i(S_k)$ informs each potential tracker T_x in TG_{sub} of its corresponding segments $B_{sub}(S_k)$. While receiving $B_{sub}(S_k)$, T_x reflectively knows $H_{sub}(S_k)$ in D_x .
- 3) T_x performs the same procedure as mentioned above to determine $H^*(B_{sub}(S_k))$ and a potential substitute p_s^x in D_x . $T_i(S_k)$ picks out p_s^* from all potential trackers' reports of their potential substitutes.

Note that the tracker-oriented and the peer-oriented method have the same effect of finding p_s^* although they have different messaging and computation overheads. The following section will examine their relative performance.

Step 3: When a p_s^* is determined, $T_i(S_k)$ tries to reclaim some upload bandwidth to resolve the downloading blocking by using p_s^* to replace some peer p_r in $H(S_k)$.

Specifically, $T_i(S_k)$ knows all segments owned by p_s^* , denoted as $O(s)$. $T_i(S_k)$ checks every segment S_s in $O(s)$: if S_s is included in $B(S_k)$, at least one peer in $H(S_k)$ is currently uploading S_s to another peer in the system. In this case, $T_i(S_k)$ collects a set of client peers that are downloading the particular S_s , which simultaneously belongs to both $O(s)$ and $B(S_k)$, from those peers in $H(S_k)$. Among these client peers, $T_i(S_k)$ chooses the one with the highest peer contribution as a "passed" client peer. Peer contribution is considered first because a passed client peer may obtain better service after changing its server peer to the substitute p_s . In case a p_s 's grade is less than its original server peer, the server may alternatively check another p_s with lower C_s , but higher G_s .

Step 4: $T_i(S_k)$ determines a passed client peer p_x and redirects it to download an indicated segment from p_s^* instead of the original server peer p_r .

Step 5: p_r thus regains some uploading bandwidth, and eventually begins uploading S_k to p_i .

IV. PERFORMANCE RESULTS

This section describes the simulation and the relative performance of the tracker-oriented and the peer-oriented distributed pairing approaches in large-scale, distributed P2P networks.

A. Simulation Environment

The experiments involve a simple simulator based on a discrete and slotted time-based model. The simulator runs a distributed model containing $n=100000$ peers evenly scattered in $TG=1, 2, \dots, 128$ network domains. Every peer arbitrarily contains one of $d=820$ files as its file sharing base. Every file has the same size $f=30,000$ KB and consists of 10 file segments of equal size $m=3000$ KB. Every peer has uniform upload and download capacities μ_i assigned by a normal distribution with $\mu_i=100$ and $\sigma=1$ or a random distribution with $\mu_i=[50, 150]$. The minimal threshold of remaining upload/download bandwidth is set as μ , that is a percent of μ_i .

The simulation runs in 8000 time units, and peers are free to join and leave the system. New peers join at a random probability $p_{in}=0.005$. A peer can leave the system at two different probabilities: $p_{out}=0.002$ when it is in active/sleep/busy/normal state, or $p_{out}=0.003$ in the leisure/idle states. For traffic generation, this simulation considers a heavy request workload. Every peer issues a new request to access a file at a random probability $p_r=0.005$. Depending on file segmentation, every peer generates 10 segment requests to perform concurrent uploadings/downloadings in a batch. Each peer is assigned a request queue of length $Q=30$ to keep its pending requests. A peer's contribution is increased by (1), in response to its segment uploading, as α is set in 0.5 ± 0.15 .

The distributed pairing scheme is inspected in terms of average messaging overhead and blocking rate.

- *Average number of outward messages per request* indicates the average of outward messages among trackers, caused by any request in a distributed P2P model.

Algorithm DP: Distributed Pairing Scheme

```

1: Procedure Do_DP_OptimalSubstitute (  $S_k, H(S_k), T_i(S_k)$  )
2:    $B(S_k) \leftarrow \text{findSegmentSet}( H(S_k) );$ 
3:   if TrackerOrientedMethod is True then
4:      $t \leftarrow \text{getNumOfTrackers}( TG );$ 
5:     for  $T_i \in TG$  and  $t \leftarrow 1, T$  do
6:        $p'_s \leftarrow \text{findPotentialSubstitute\_CBF}( H(S_k), B(S_k), T_i, D_i );$ 
7:        $\triangleright$  CBF(): to find out a local substitute peer
8:        $H^*( TG ) \leftarrow H^*( TG ) \cup p'_s;$ 
9:     end for
10:  else  $\triangleright$  PeerOrientedMethod
11:     $n \leftarrow \text{getNumOfPeers}( H(S_k) );$ 
12:    for  $p_n \in H(S_k)$  and  $n \leftarrow 1, N$  do
13:       $L_{<S,T>} \leftarrow \text{getListOfPotentialTracker}( p_n );$ 
14:       $TG_{sub} \leftarrow TG_{sub} \cup L_{<S,T>};$ 
15:    end for
16:     $t \leftarrow \text{getNumOfTrackers}( TG_{sub} );$ 
17:    for  $T_i \in TG_{sub}$  and  $t \leftarrow 1, T$  do
18:       $p'_s \leftarrow \text{findPotentialSubstitute\_CBF}( H_{sub}(S_k), B_{sub}(S_k), T_i, D_i );$ 
19:       $H^*( TG ) \leftarrow H^*( TG ) \cup p'_s;$ 
20:    end for
21:  end if
22:  if  $H^*( TG ) \neq \{\emptyset\}$  then  $\triangleright$  Enforcing Peer Substitute Policy
23:     $H^*( TG ) \leftarrow \text{sortGrade}( H^*( TG ), \downarrow );$ 
24:     $p^*_s \leftarrow \text{getFristPeer}( H^*( TG ) );$   $\triangleright p^*_s$ : an optimal substitute
25:     $O(s) \leftarrow \text{findSegmentSet}( p^*_s );$ 
26:     $M \leftarrow \text{getNumOfSegments}( O(s) );$ 
27:    for  $S_s \in O(s)$  and  $s \leftarrow 1, M$  do
28:      if  $S_s \in B(S_k)$  then
29:         $H(S_s) \leftarrow \text{findPeerSet}( S_s, H(S_k) );$ 
30:         $H(S_s) \leftarrow \text{sortContribution}( H(S_s), \downarrow );$ 
31:         $p_r \leftarrow \text{getFristPeer}( H(S_s) );$   $\triangleright p_r$ : a passed client peer
32:         $p_r \leftarrow \text{findServerPeer}( p_r, S_s );$ 
33:         $\text{setServerPeer}( p_r, p_s, S_s );$   $\triangleright$  Set  $p_s$  as  $p_r$ 's new server peer
34:         $\text{startDownload}( p_r, S_s );$ 
35:         $\text{startDownload}( p_r, p_r, S_k );$ 
36:        break;
37:      end if
38:    end for
39:  else  $\triangleright$  Enforcing Peer Elimination
40:     $H(S_k) \leftarrow \text{sortContribution}( H(S_k), \downarrow );$ 
41:     $p_{low} \leftarrow \text{getLastPeer}( H(S_k) );$ 
42:    if  $p_i$ 's  $C_i > p_{low}$ 's  $C(low)$  then
43:       $p_r \leftarrow \text{findServerPeer}( p_{low}, S_k );$ 
44:       $\text{stopDownload}( p_{low}, S_k );$ 
45:       $\text{startDownload}( p_i, p_r, S_k );$ 
46:    else  $\triangleright p_r = \text{null}$ 
47:      Unsuccessfully!  $p_i$ 's request for  $S_k$ ;
48:    end if
49:  end if
50: end procedure

```

Fig. 2. Algorithmic form.

- *Average ratio of pending requests* is the total number of pending requests in queue to the total of segment requests that all peers have sent to the system.

B. Experimental Results

This study simulates the distributed pairing scheme and compares tracker-oriented and peer-oriented methods in a distributed P2P environment. Experiments were conducted to examine the influences of different factors, including tracker group size, initial number of segments owned by each peer, bandwidth granularity, and peer contribution. The experimental results in Fig. 3(a)-(d) show some observations below.

First, the tracker-oriented method induces a linear growth in message overhead as the tracker group size increases, as Fig. 3(a) shows. In contrast, the message overhead generated by the peer-oriented method remains nearly constant regardless of the tracker group size. This is because the hosting

tracker in the tracker-oriented method forwards $H(S_k)$ to all external trackers and asks them to find an optimal substitute server peer. This process propagates a considerable amount of messages across the tracker network. In the peer-oriented method, however, the hosting tracker first determines a subset of potential trackers. This process avoids a “blanket” search in the tracker network.

Second, computing a pre-determined subset of potential trackers using the peer-oriented method is cost-effective, since it efficiently reduces the overall message overhead. Though this pre-determining process generates some computation and communication overheads, these costs are limited in the hosting tracker's domain. Hence, this process is superior to the high expense of the tracker-oriented method. The additional experimental results depicted in Fig. 3(c) support this observation in that the tracker-oriented method sends more outward messages to external trackers than the peer-oriented method.

Third, bandwidth granularity directly influences the measure of message overhead. Both peer-oriented and tracker-oriented methods produce lower message overheads under higher granularity. For instance, the obvious differentiation between the cases of $\mu=10\%$ and $\mu=50\%$ depicted in Fig. 3(a) confirms this observation.

Fourth, the tracker-oriented method is sensitive to the variance of number of segments that every peer begins with. The greater the initial number is, the smaller the number of messages a request may incur. When the initial number is larger enough, the average number of outward messages caused by every request approaches zero. In other words, the hosting tracker handles almost every request in its local domain. Adding more initialized segments effectively increases the probability of finding a server peer within the domain. On the other hand, the message overhead of the peer-oriented method remains low for the same reason as mentioned above.

Finally, using the tracker-oriented or the peer-oriented method achieves the same result of finding an optimal substitute server peer p^*_s in a large-scale distributed P2P network, though they generate different message overheads. Both the peer-oriented and tracker-oriented methods can augment service fairness by peer substitute and elimination. Figure 3(d) depicts the measures of blocking rates among peer clusters. The blocking rate is very low when no server peers are available as a result of temporary unavailability due to unsuccessful peer elimination. Again, the blocking rate can be much reduced as the system has segments scattered over a distributed network.

The discussion above reviews the peer-oriented and tracker-oriented methods in a distributed P2P network environment. Based on the relative performance indicated in Fig. 3, the peer-oriented method is superior to the tracker-oriented method in terms of efficiency and cost-effectiveness. Nevertheless, whereas the peer-oriented method induces a higher computation overhead, the tracker-oriented method is preferable under some critical situations. For example, peers may be resource-constrained to reduce computation cost, or when transmission between the tracker and its mobile or thin peers is limited or unreliable. Hence, though the tracker-oriented method is not

economical, it is a viable alternative under certain conditions.

V. CONCLUSION

This study presents a distributed pairing scheme to addresses the peer churn and free riding problems appearing in large-scale distributed P2P networks. This design develops and compares two variants of peer-oriented and tracker-oriented pairing methods. Performance results show that both methods maintain the effects of specific peer substitute and elimination policies. However, the peer-oriented method outperforms the tracker-oriented method in terms of message overhead, but may induce higher computation costs. Hence, these two methods are alternatives with different communication and computation concerns. Consequently, the proposed distributed pairing scheme is able to ensure fast and fair file sharing applications in large-scale P2P networks. Extensive simulations confirm the efficacy, applicability, and extensibility of the proposed scheme.

REFERENCES

- [1] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, December 2004.
- [2] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *Proceedings of IEEE INFOCOM'04*, vol. 4, March 2004, pp. 2242–2252.
- [3] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Tutorials and Surveys*, vol. 7, no. 2, pp. 72–93, July 2005.
- [4] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, October 2006, pp. 189–202.
- [5] C.-L. Hu and T.-H. Kuo, "Hierarchical peer-to-peer overlay with cluster-reputation-based adaptation," in *Proceedings of the 2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, August 2009.
- [6] F. E. Bustamante and Y. Qiao, "Designing less-structured p2p systems for the expected high churn," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 617–627, June 2008.
- [7] E. Adar and B. A. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, no. 10, October 2000.
- [8] A. Creus-Mir, R. Casadesus-Masanell, and A. Hervas-Drane, "Bandwidth allocation in peer-to-peer file sharing networks," *Computer Communications*, vol. 31, no. 2, pp. 257–265, February 2008.
- [9] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, issue. 12, pp. 45–48, December 2000.
- [10] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," in *Proceedings of IEEE INFOCOM'03*, vol. 3, April 2003, pp. 2188–2198.
- [11] C.-L. Hu, Y.-H. Chang, D.-Y. Chen, and Y.-W. Chen, "Simple peer selection strategies for fast and fair peer-to-peer file sharing," in *Proceedings of the 12th International Conference on Advanced Communication Technologies (ICACT'10)*, February 2010.
- [12] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *ACM SIGecom Exchanges*, vol. 5, no. 4, pp. 41–50, July 2005.
- [13] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proceedings of ACM SIGCOMM'04*, August 2004, pp. 367–378.

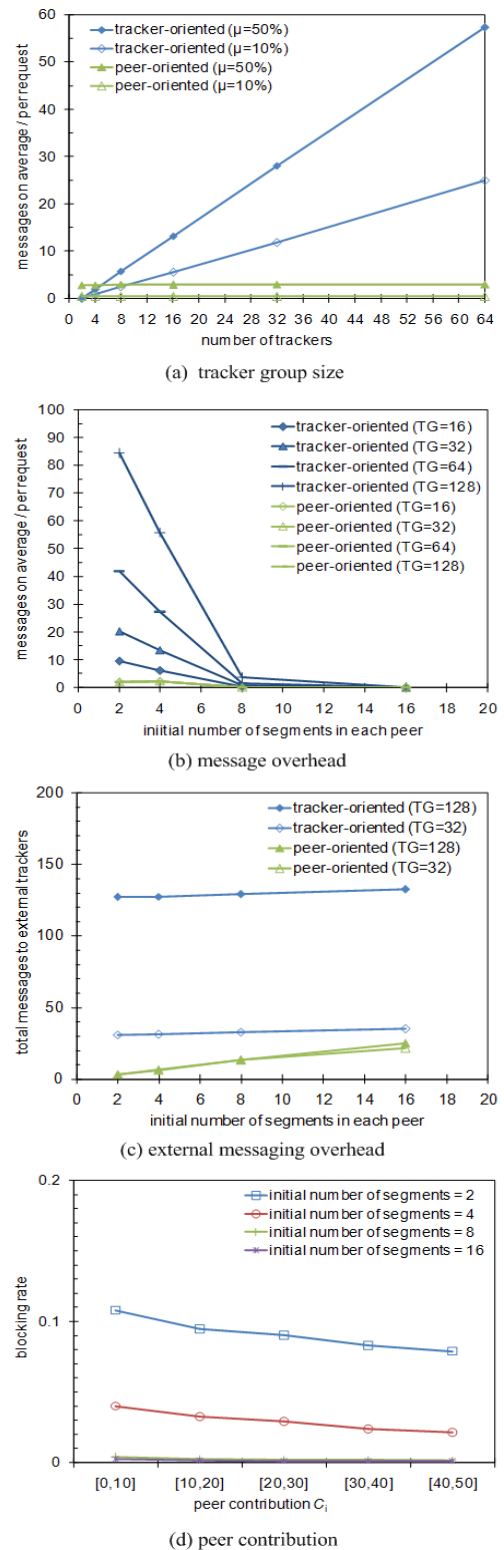


Fig. 3. Performance results with $n=100000$, $TG=[2, 4, \dots, 128]$, $p_r=0.005$, $\alpha=0.5$, $t=1000$, and $\mu=[10, 30, 50\%]$: (a) average number of outward messages per request against tracker group size (as one initial segment is set), (b) average number of outward messages per request against the initial number of segments in each peer (as $\mu=30\%$), (c) total of outward messages across the tracker network against the number of initial segments in each peer (as $\mu=30\%$ and $TG=128$), and (d) block ratio in a peer cluster against the number of initial segments in a peer (as $\mu=30\%$ and $TG=128$).