Fair Peer Assignment Scheme for Peer-to-Peer File Sharing

Chih-Lin Hu[1], Da-You Chen[1], Yi-Hsun Chang[1] and Yu-Wen Chen[2]
[1] Department of Communication Engineering, National Central University
Taoyuan, Taiwan 32001 – R.O.C.
[e-mail: clhu@ce.ncu.edu.tw; {955003009, 975203043}@cc.ncu.edu.tw]
[2] Department of Electrical Engineering, Columbia University
New York, NY 10027 – USA.
[e-mail: cheneva00@gmail.com]
*Corresponding author: Chih-Lin Hu

Received July 9, 2010; accepted August 20, 2010;
published October 30, 2010

Abstract

The reciprocal virtue of peer-to-peer networking has stimulated an explosion of peer population and service capacity, ensuring rapid content distribution in peer-to-peer networks. Critical issues such as peer churn, free riding, and skewed workload significantly affect performance results such as service agility, fairness, and resource utilization. To resolve these problems systematically, this study proposes a peer assignment scheme that supports fair peer-to-peer file sharing applications. The proposed scheme exploits the peer duality of both server-oriented peer capacity and client-oriented peer contribution. Accordingly, the system server can prioritize download requests and appropriately assign server peers to uploading file objects. Several functional extensions, including peer substitution and elimination, bandwidth adjustment, and distributed modification, help cope with subtle situations of service starvation and download blocking, and hence make the system design robust and amenable. Simulation results show this design is examined under both centralized and distributed peer-to-peer environments. Performance results confirm that the proposed mechanisms are simple but effective in maintaining service agility and fairness, without loss of overall service capacity in peer-to-peer files sharing systems.

Keywords: Peer assignment, peer management, file sharing, content distribution, P2P

This article is aimed at supplanting the previous conference version, presented in Proceedings of the 12th Int'l Conf. on Advanced Communication Technologies (ICACT 2010), Phoenix Park, Korea, February 7-10, 2010. This article contains fully materials of extended scheme design and development, algorithmic forms, newly extensive performance evaluation, and a review of related works. This work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC98-2221-E-008-041 and NSC99-2221-E-008-011.

DOI: 10.3837/tiis.2010.10.002

1. Introduction

Peer-to-peer (P2P) networking technology has recently emerged as a new variant of the distributed computing paradigm for building distributed networked applications. With the proliferation of P2P applications, a major portion of today's Internet traffic is generated by P2P file sharing applications [1]. P2P networking differs from traditional client/server networking in several aspects, including peer duality, service capacity, peer churn, overlay organization, free riding, etc. Perhaps the intrinsic difference is *peer duality*, in that each peer plays a dual role of both service provider (server) and a consumer (client) of the implemented service [2]. In many P2P file sharing applications [3], like Napster, Gnutella, KaZaA, Morpheus, eDonkey, and BitTorrent, a peer requests files from its peers, and also stores and serves files to its peers. Increasing the peer population not only increases the workload, but also produces a concomitant increase in service capacity to process the request workload [4]. In a traditional network, however, clients and servers are distinct. An increase in the client population simply increases the workload, inducing the scalability problem and degrading performance. Thus, peer duality distinguishes P2P networking from traditional client/server networking.

Peer *churn*, which represents the dynamics of peer participation in a system[1], significantly influences overlay design, resiliency, and assessment [5][6]. Peers in a system cooperate to run an application-level overlay network [7] that provides connectivity, signal messaging, routing, discovery, and searching between end hosts that are addressable on top of IP communication networks. An ideal P2P overlay is autonomous, with self-organization and self-management against peer churn. An ideal P2P overlay is also resilient, with fault detection and repair facilities. However, due to peer autonomy and mutual dependency, peer transiency and its implications have a great effect on the efficacy of replication, search, and query mechanisms in content distribution [8][9].

The problem of free riding in a P2P system negates the reciprocal provision of resources among peers. This problem is separate from the concept of traditional distributed applications. Free riding is irrelevant to P2P networking, but inherits from P2P participants' attitudes of mind. A large percentage of the peer population consists of *free riders* - maximizing their own utility disproportionate to their contribution to the system [10][11]. Unfortunately, free riders take advantage of generous peers by consuming service capacity. If no request admission or scheduling policy is enforced to maintain service fairness, generous peers often become frustrated and refuse to contribute anymore [12]. Some studies propose *incentive* approaches to encourage peer contribution and guarantee service fairness [13][14]. The growing recognition of reputation systems has generated significant research in this area, as Section 2 shows. Reputation-based schemes determine service priorities based on the histories of peer contributions to the system [15]: reputable peers receive high priority and are rewarded more resources to compensate for their contribution.

This study systematically addresses the issues of peer churn, free riding, and workload dynamics, and considers the dual factors of server-oriented peer capacity and client-oriented peer contribution in a P2P computing context. This study designs a fair peer assignment scheme for file sharing in a dynamic P2P network environment. The proposed scheme

[1] A peer's lifespan is transitory because a peer can freely join in the system, be active for some time, and then go off-line, unpredictably removing itself from the system.

maintains service agility and fairness among peers in response to peer churn and free riding. The basic concept here is to gradate client-oriented peer contribution and server-oriented peer capacity based on uploading capacity and downloading bandwidth, respectively. The system server performs peer management that records peers' uploading and downloading behavior. Based on request workload, the system server applies specific peer assignment strategies to control the request admission and scheduling processes to guarantee fair and efficient use of resources without decreasing overall system throughput. Therefore, the results of this study are summarized as follows.

- This study formulates the measures of peer contribution and capacity. In light of indirect-reciprocity notion [13], a basic peer assignment strategy (PAS) is designed to choose server peers to process download requests in descending order of peer contribution.
- The advanced peer assignment strategy (APAS), which extends PAS with a specific substitute policy, can modify peer assignment in response to the traffic dynamics created by peer churn and varying request workload.
- The APAS is associated with a specific peer elimination process (APAS-E) to cope with possible service starvation and further boost the ability of service differentiation and fairness.
- The extended design of a distributed APAS (D-APAS) is applicable to large-scale partially-centralized and structured P2P environments [3]. This design includes two kinds of tracker-oriented and peer-oriented D-APAS schemes that maintain the functionalities and effects mentioned above.
- The experimental simulations in this study assess performance sensitivities in terms of several measure metrics, average download time, ratio of pending requests, download count, and download ratio by contribution range. This study also compares the two kinds of D-APAS in terms of communication overhead.

Consequently, this study presents a systematic methodology that provides a basic peer management mechanism and an advanced peer assignment strategy with several auxiliary functions. The proposed design protects system throughput, service agility, and fairness against service starvation and varying traffic dynamics in P2P networks. The proposed design is suitable for both centralized and distributed P2P file sharing systems.

The rest of this article is organized as follows. Section 2 reviews a number of related studies on P2P file sharing applications. Section 3 describes the P2P system model. Section 4 details the proposed PAS, APAS, APAS-E, and D-APAS. Section 5 presents the performance results, while Section 6 provides the conclusion.

2. Related Work

The astounding growth of P2P file sharing applications has accelerated the spread of multimedia contents, typically music and video categories, throughout the Internet. The success of the P2P paradigm is dependint on its users' altruistic provision and reciprocal cooperation in sharing storage and computation resources. However, peers in a P2P system have different resources, capabilities, and user attitudes. The egoistic users, i.e., free riders that exist in any P2P system [10], make a utopian P2P society impossible.

The free riding problem – which is not a networking or system issue, but originates from user behavior – is difficult to resolve in P2P systems [16][17]. Rational participants may refuse to contribute their fair share of resources, and only seek to maximize their own utility.

Thus, individual rationality conflicts with social welfare. As reported in [11], Gnutella and Napster systems have many free riders, who represent about 70 % all peers.

Different incentive approaches encourage peers to contribute to the system [13]. Monetary payment schemes dictate that service recipients must pay service providers for the resources they consume [18]. Many P2P file-sharing systems use the game theory, e.g., the Nash equilibrium [19], to study the potential benefits of micro-payment methods [20]. However, this scheme has the critical premise of requiring an accounting and micro-payment infrastructure, which is impractical in an open P2P network [21][22]. Reciprocity-based schemes use the histories of peers' contributions to the system as decision making processes [15], and map peers' reputation scores to admission and resource allocation strategies [17][23]. Reputable peers enjoy higher priority and are rewarded with more resources to compensate for their contributions. Previous research [3][7] shows that the growing recognition of reputation systems [15] has created a significant amount of research on this topic. The following discussion review some of the incentive mechanisms in BitTorrent and many other P2P applications.

In reciprocity-based schemes, peers maintain the histories of the past behavior of other peers and use this information in their decision making processes. These schemes can be based on direct reciprocity or indirect reciprocity. Direct-reciprocity schemes are appropriate for applications with long session durations, as they provide ample opportunities for exchanging information between pairs of peers. Typical examples are BitTorrent-like applications that employ the tit-for-tat incentive mechanism [24], in which a peer only reciprocates to peers that allow it to download; in this case, uploading is a part of the protocol in the rarest first and choke algorithms [25]. Nevertheless, BitTorrent still has inherent choking issues. Though it can be moderated by the associated optimistic unchoking method which randomly probes a new connection to upload, the increase of free riders strikes a trade-off, as reported in [26]. Many studies [20][27][28][29][30] are dedicated to tackle the free-riding problems in BitTorrent.

Indirect-reciprocity schemes, a.k.a. reputation-based schemes [15][23][31][32], differ from direct-reciprocity schemes not only in computing reputation score/credit of how much a peer has contributed to the system, but also in the mapping of scores to different admission and allocation strategies. Reputation scores among peers can be pre-computed based on the history of peer behavior, and maintained by any reputation system that performs in an out-of-band manner. Reputation-based schemes are relatively robust against misbehaving or malicious peers, and sustainable for P2P systems with large peer populations, highly dynamic memberships, and infrequent repeat transactions. Through reputation differentiation, reputable peers have the right of priority service. Exmples include better quality of service and user experience, prioritized download capacity, or fast query/response. A punishment policy may be imposed on disreputable peers using different punitive levels based on the free-riding severity. For example, this policy may limit the propagation of their messages, ignore their queries, or even disconnect malicious or unproductive peers from the network [17].

On the other hand, query or search mechanisms are important in a P2P system. In highly structured P2P systems, e.g., DHT and Chord [7], seraching for an object is strictly based on its logical object identifier, and is usually done in $log(n)$ steps. However, protecting this structure from peer churn is costly and incurs a significant overhead, when the system is highly dynamic. Moreover, structured P2P systems have difficulty of implementing complex queries, such as keywords-based search and regular expression-based search. In contrast, less structured or unstructured systems have lower maintenance complexity and can adapt to node heterogeneity and network dynamics. These systems locate files independent of the system

topology by resorting to "near blind" query strategies such as flooding, random walks, *k*-random-walks, iterative deepening, breadth-first search (BFS), and deepth-first search (DFS) techniques [33][34]. However, there are two problems with these search mechanisms. On one hand, a peer often receives duplicate query messages. This highlights the need to study potential approaches to reducing search overhead and query response time [35][36][37][38]. On the other hand, when a high-degree peer maintaining P2P connectivity is overloaded, queries should be directed toward other peers by taking server peer capacity into account.

Peer duality, which means that each peer is both a *client peer* and a *server peer*, is a special feature that distinguishes the P2P file sharing paradigm from other distributed systems. Reciprocal provision among peers effects the spread of file replications, increases the file volume, and improves the service capacity of the network. Previous studies address how the free riding problem affects service unfairness. Though free riders may not reduce system throughput, they may unfairly occupy or utilize system resources [2]. The traced-based analysis in [4] shows that service capacity, i.e., average throughput, in a P2P network experiences exponential growth until it reachs the steady state. The sensitivity of this growth is practically related to file segmentation, peer selection, access admission and scheduling policy, and traffic dynamics.

Previous studies [2][4] serve as the motivation for this study to develop a simple peer assignment strategy that considers the dual factors of peer contribution and capacity in conventional P2P systems. The basic strategy aims to schedule download requests based on peer contributions and assign appropriate server peers in charge based on peer capacity. Several auxiliary functions sustain performance against service starvation and traffic dynamics [9]. This study further applies the proposed strategy, with uncomplicated modifications, in distributed P2P systems, and achieves the same effect. Therefore, this study presents an advanced peer assignment strategy capable of guaranteeing service agility and fairness in dynamic P2P environments.

3. System Modeling

This section describes a conventional P2P system environment, and specifies the bandwidth resource utilization and peer state management used for resource allocation and content distribution in P2P file sharing applications.

3.1 System Environment

Fig. 1 shows a centralized, structured P2P system environment [3] on which this proposal is based. Specifically, there is a central server that performs the functions of tracking, peer assignment, and service provision to support distributing file segments within a P2P network. A tracker server is located in a central server or another dedicated host attached to a central server[2]. This tracker server maintains a segment meta-data repository and performs segment management and peer management in the system. **Fig. 2** shows that a tracker server provides the volumes of segment meta-data records, each of which includes a segment index, available location references, and other attributes. For every segment, the tracker records a set of peers that own this segment, and manages these peers by referring to their *states* in service. Thus, every peer connects to the central server and inquires about where any indicated segment can be downloaded. The server promptly processes data access to the meta-data repository, and

[2] Consider that a central server is coupled with a dedicated hosting tracker inside a P2P domain. Without ambiguity, the roles of a hosting server and its tracker can be used interchangeably in the rest of this article.

then selects one out of many server peers according to a specific peer assignment strategy. With a request reply, a client peer directly connects to its assigned server peer to access the indicated segment. Then, with a pair of client and server peers, the central server can monitor the download/upload sessions and adjust bandwidth allocation in a timely manner. In addition, the server peer can record the download state of every requested segment, remaining upload bandwidth, and uploading contribution into its property profile.

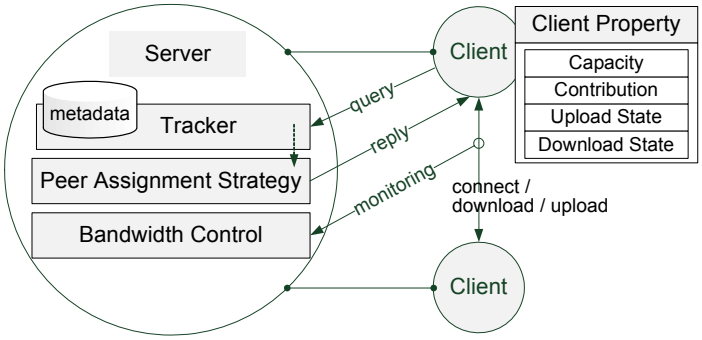


Fig. 1. A centralized, structured P2P file sharing context.

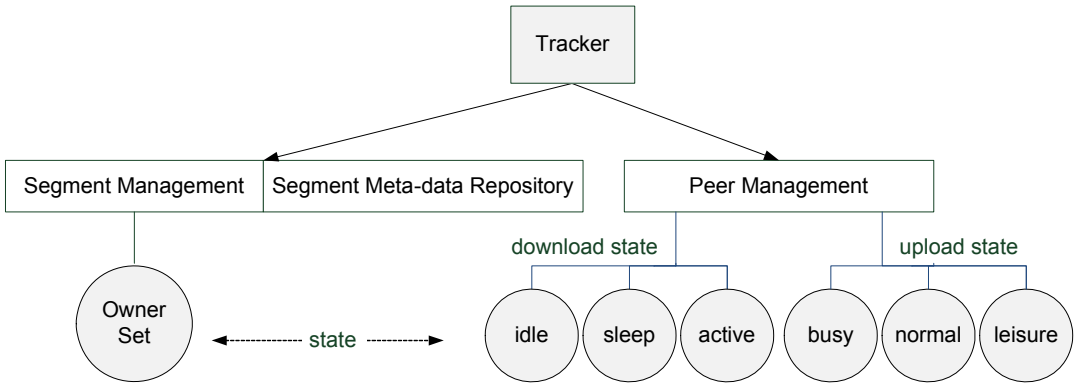


Fig. 2. A tracker functional reference.

3.2 Resource Utilization

The P2P system enforces the premise that the granularity of upload and download bandwidth utilization is based on a slotted time model. 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. This precondition facilitates the analysis of bandwidth granularity.

Selecting a *fit* server peer is a decisive step before a client peer starts to download any segment. The system performs peer management based on bandwidth availability. Specifically, the following definitions provide the update and download states of each peer.

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.

Table 1. Symbols used in the peer assignment strategy.

Symbol	Meaning
μ	the low bound of available data transfer rate for uploading or downloading a segment
n	the amount of peers in the system
P_i	peer identifier, for i=1, 2, 3,…, n
S_k	segment identifier, for i=1, 2, 3,…, m
r^i_k	the request for downloading S_k sent by a peer P_i
C_i	the contribution of a peer P_i for file segment distribution
$UL_{past}(i)$	the number of segments which a peer P_i had uploaded in the past
$UL_{now}(i)$	the number of segments which a peer P_i is uploading now
$R(i)$	the set of segments which a peer P_i is asking for in response to its pending requests
$H(S_k)$	the set of peers which have S_k within a specific tracker's domain
$B(S_k)$	the set of segments which peers in $H(S_k)$ are delivering
$H^*(B(S_k))$	the candidate set of substitute server peers which have at least a segment belonging to $B(S_k)$
G_i	the average upload bandwidth which a server peer P_i can now offer
$O(s)$	the set of segments which a substitute peer P_s owns
μ_i	the total upload/download bandwidth which P_i owns
$I(S_k)$	the relative immediacy of segment S_k to be uploaded first in the system
D_i	a P2P domain identifier, for i=1, 2, 3,…, t
T_i	a tracker identifier, for i=1, 2, 3,…, t
TG	the group of all trackers in the system as considered
$T_i(S_k)$	the corresponding/hosting tracker T_i which handles the request r^i_k
$H^*(TG)$	the set of potential substitute server peers reported from trackers in TG
P^*_s	the optimal one out of substitute server peers in the system

4. Peer Assignment and Management Strategies

This section consists of two parts. The first part proposes peer assignment strategies in a centralized P2P model, and the second adapts these designs to provide distributed versions in a large-scale P2P model. Section 4.1 formulates the measures of a client peer's contribution and server peer's grading of bandwidth capacity, which are both used to differentiate request priority and assignment among peers. Section 4.2 designs a basic peer assignment strategy (PAS) in a centralized P2P model. Section 4.3 presents an advanced peer assignment strategy (APAS) that extends the PAS with a substitute policy for peer assignment. Section 4.4 integrates APAS with a peer elimination and bandwidth reclamation method (APAS-E) to guarantee fair bandwidth allocation. Furthermore, Section 4.5 remodels the APAS and APAS-E as D-APAS. Section 4.6 describes a bandwidth adjustment method that can be incorporated into APAS or D-APAS to cope with asymmetric bandwidth utilization. Finally,

this study presents a joint design of peer assignment and management mechanism to protect a P2P system from peer churn and free riding.

To ease of exposition in this section, **Table 1** lists the meanings of symbols used in the proposed scheme. These denotations are identical in both centralized and distributed P2P systems unless special notes are given to discern the working domain that corresponds to a specific hosting server/tracker.

4.1 Peer Contribution and Grading

The proposed peer assignment and management strategy adopts an incentive-based approach to encourage peers to contribute their own resources to achieve better performance. Notice that the direct-reciprocity and tit-for-tat incentive approaches in BitTorrent-like applications have inherent choking issues [26]. The proposed scheme adopts an indirect-reciprocity-based incentive approach that 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:

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

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 two terms. The terms UL_{past} or UL_{now} 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. Equation (1) shows that the central server prioritizes download requests from client peers.

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

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

4.2 Basic Peer Assignment Strategy (PAS)

The PAS considers both service fairness and agility during peer selection. Basically, PAS appoints server peers to take the request workload in descending order of peer grading, and schedules requests in descending order of peer contribution. PAS sets a minimal bound μ of available upload and download bandwidth to improve download speed. A request from a client peer of the highest C_i is handled first. Then, the server peer with the highest G_i deals with this request. Thus, PAS can simply and evenly balance the use of upload bandwidth capacities among peers and avoid bandwidth fragmentation throughout the system.

Fig. 3 illustrates the PAS procedure. Given a peer P_i whose download bandwidth is higher than μ, P_i sends the central server a request r^i_k to download a segment S_k. With a r^i_k, the tracker generates a peer set, called $H(S_k)$, that includes all peers having S^i_k in their local storages. Then, the server selects the server peer P_r with the highest upload rate in $H(S_k)$ using PAS. If P_r's upload bandwidth is not lower than μ, the server notifies P_i of its server peer P_r in charge. P_i then negotiates with P_r for downloading S_k. If P_r does not have enough upload bandwidth to serve this request at this moment, the server will mark P_i's download state as sleep and push this request back into the request queue. The server later checks if r^i_k is valid yet. Algorithm 1 presents the PAS's algorithmic form for reference.

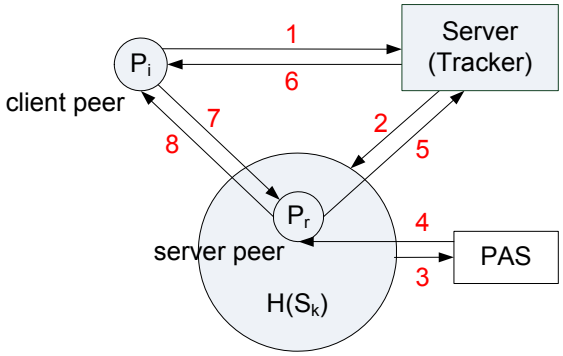


Fig. 3. A basic peer assignment strategy.

Algorithm 1: Basic Peer Assignment Strategy

Input: $R(i) \neq \{\varnothing\}$, $H^*(B(S_k)) = \{\varnothing\}$, P_r = **null**
Output: P_r : P_i's designated server peer
Begin
$R(i) \leftarrow$ *getRequestSet*($R(i)$);
$K \leftarrow$ *getNumOfSegments*($R(i)$);
for $S_k \in R(i)$ and k $\leftarrow$ *1*, K **do**
  $H(S_k) \leftarrow$ *findPeerSet*(S_k , *AllPeers*);
  $H(S_k) \leftarrow$ *sortGrade*($H(S_k)$, $\downarrow$);
  $P_r \leftarrow$ *getFirstPeer*($H(S_k)$);
  if *Grade*(r) $\geq \mu$ **then** ▷ P_r's grading
    startDownload(P_i , P_r , S_k);
  else ▷ APAS Strategy – Substitute policy
    doAPAS_SubstitutePolicy(S_k , $H(S_k)$); ▷ Refer to Algorithm 2
  end if
end for
End

4.3 Advanced Peer Assignment Strategy (APAS)

The APAS modifies PAS with functional extensions to support "concurrent" and "non-blocking" downloading mechanisms. The tie-in "substitute policy" on peer assignment not only sustains fairness and throughput, but also alleviates the service starvation problem due to dynamic changes of skewed access workload and peer churn in a dynamic P2P context.

4.3.1 Concurrent Downloading

Given a number of pending requests from a peer P_i in queue, the central server has a set of all segments requested by P_i, denoted as $R(i)$. As in PAS, for each S_k in $R(i)$, the server quickly finds a server peer from $H(S_k)$ for uploading S_k. If $R(i)$ contains more segments, this process repeats itself until P_i's remaining download bandwidth is less than μ. The processing sequence of requests in $R(i)$ may depend on any specific schedule, such as first-in-first-out (FIFO), earlier deadline first (EDF), high download speed first, etc. However, the analysis of request scheduling methods is orthogonal to this study. For simplicity, assume that every request is subject to the same processing deadline. The proposed scheme adopts the FIFO simply as a development baseline.

4.3.2 Non-blocking Downloading

In PAS, there is a critical situation of download blocking due to "service starvation." This situation - with no uploading of any segment S_k in $R(i)$ - 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, especially when *traffic dynamics* is considered. When all peers owning a segment S_k in $R(i)$ are in the busy state, they have no more bandwidth to upload. All requests for $R(i)$ are then blocked until any P_r in $H(S_k)$ reclaims sufficient upload bandwidth to accept a new request. This increases the likelihood of PAS failure due to service starvation, resulting in poor performance. The next subsection presents a peer substitute technique to cope up with this situation.

4.3.3 Substitute Policy

The peer assignment scheme employs a peer substitute policy to address the possibility of service starvation. A substitute policy attempts to find a new substitute peer P_s 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. Note that this policy can alleviate the service starvation with respect to many influential factors under dynamic traffic.

The following steps specify this substitute policy and its procedure (Steps 1-5) in reference to **Fig. 4**. To ease exposition, Algorithm 2 shows the algorithmic form of the APAS with a substitute policy.

Step 1: Let a request r^i_k run into a download blocking case. The central server firstly assembles a super set of segments, denoted as $B(S_k)$, which all peers in $H(S_k)$ are currently delivering.

Step 2: For every segment S_l in $B(S_k)$, the central server tries to find a candidate server peer P_l. A peer qualifies as a candidate server peer if it is the one with the highest grade G_i out of all peers that have S_l, and its grade is no less than μ. Therefore, the server has a set of candidate server peers, denoted as $H^*(B(S_k))$. The peer with the highest grade in $H^*(B(S_k))$ is then chosen as a substitute server peer P_s.

Step 3: When a P_s is determined, the central server tries to reclaim some upload bandwidth to resolve the download blocking problem by using P_s to replace some peer in $H(S_k)$. Specifically, the server knows all segments owned by P_s, denoted as $O(s)$. The server 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, the server 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 $H(S_k)$.

Step 4: Among these client peers, the central server 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 5: The server instructs a passed client peer to download S_s from P_s instead of its original server peer in $H(S_k)$. Then, the original peer regains some uploading bandwidth and eventually commences uploading S_k to P_i.

Fig. 4 illustrates the usage of the substitute policy. Let $H(S_k)$ include two peers, P_1 and P_2. Initially, both P_1 and P_2 are in the busy state. P_1 is delivering S_1 to P_x and S_2 to another peer, and P_2 is delivering S_3 to P_y and S_4 to another peer. $B(S_k)$ now includes S_1, S_2, S_3, and S_4. The server accordingly determines $H^*(B(S_k))$, which includes all peers sending any segments in

$B(S_k)$. Let the server select P_s as a candidate server peer from $H^*(B(S_k))$. Since P_s has S_1 and S_3, the server checks $H(S_k)$ and finds that two client peers P_x and P_y are downloading S_1 and S_3 from P_1 and P_2, respectively. The server selects P_x or P_y depending on who has the highest contribution as the passed client peer. This passed client peer redirects its download from P_s instead of its previous server peer. Eventually, either P_1 or P_2 in $H(S_k)$ now has enough upload bandwidth to accept another pending request in the queue.

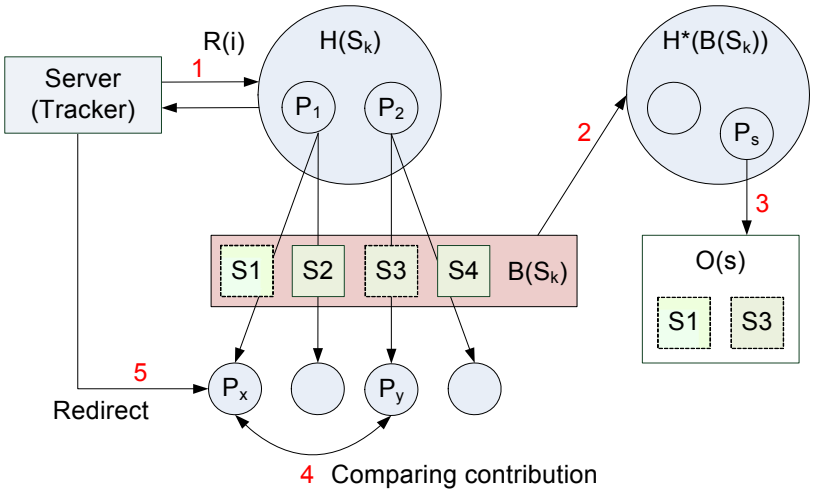


Fig. 4. The APAS with substitute policy.

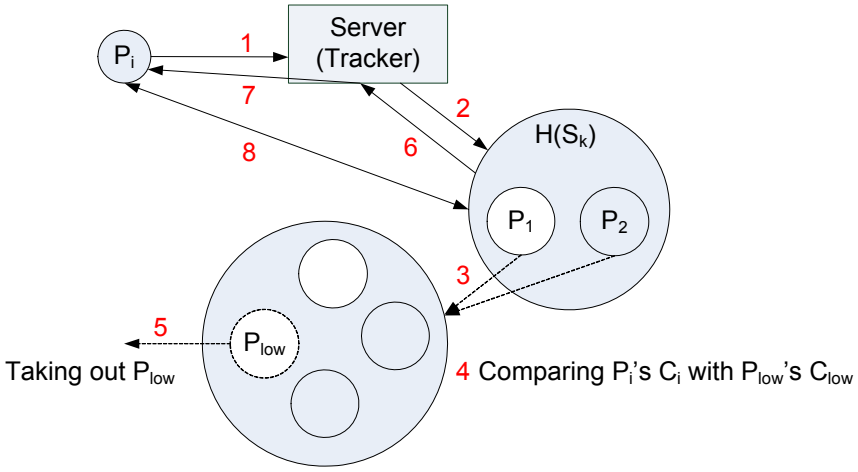


Fig. 5. The APAS with peer elimination.

4.4 APAS with Peer Elimination (APAS-E)

In APAS, there is a subtle situation in which a central server cannot find any suitable substitute server peers, since none of peers in $H^*(B(S_k))$ have enough remaining upload bandwidth. This study devises a peer elimination method to address this problem. The proposed method attempts to reduce or cancel a peer's ongoing downloading if this peer has low contribution, such as a free rider. That is, a peer with a higher contribution can preempt a downloading session for the sake of fair resource allocation.

Fig. 5 shows the procedure of APAS with peer elimination in reference to Algorithm 3. Based on the previous example in **Fig. 4**, suppose that the central server cannot find a

substitute server peer P_s to upload a particular S_k in $R(i)$. The server further checks all potential server peers in $H(S_k)$ and their client peers. Among these client peers, the P_{low} with the lowest contribution is selected to compare with P_i. If P_i's contribution is higher than P_{low}'s, P_i will replace P_{low} to use upload bandwidth from P_1, given that P_{low} downloads S_k from P_1. The server cancels P_{low}'s ongoing download, and instructs P_{low} to resend a request for S_k, or alternatively defers its download until another server peer becomes available.

Algorithm 2: Advanced Peer Assignment Strategy with Substitute Policy

```
1:  Procedure DoAPAS_SubstitutePolicy( S_k, H(S_k) )
2:    B(S_k) ← findSegmentSet( H(S_k) );
3:    l ← getNumOfSegments( B(S_k) );
4:    for S_l ∈ B(S_k) and l ← 1, L do
5:      H(S_l) ← findPeerSet( S_l , AllPeers );
6:      H(S_l) ← sortGrade( H(S_l) , ↓ );
7:      P_l ← getFirstPeer( H(S_l) );
8:      if Grade( l ) ≥ μ then
9:        H*( B(S_k) ) ←H*( B(S_k) ) ∪ P_l ;
10:     end if
11:   end for                          ▷ A candidate set of substitute peers
12:   if H*( B(S_k) ) ≠ {∅} then
13:     H*( B(S_k) ) ← sortGrade( H*( B(S_k) ) , ↓ );
14:     P_s ← getFristPeer( H*( B(S_k) ) );          ▷ P_s: a substitute peer
15:     O(s) ← findSegmentSet( P_s );
16:     M ← getNumOfSegments(O(s) );
17:     for S_s ∈ O(s) and s ← 1, M do
18:       if S_s ∈B(S_k) then
19:         H(S_s) ← findPeerSet( S_s , H(S_k) );
20:         H(S_s) ← sortContribution( H(S_s) , ↓ );
21:         P_t ← getFirstPeer( H(S_s) );          ▷ P_t: a passed client peer
22:         P_r ← findServerPeer( P_t , S_s );
23:         setServerPeer( P_t , P_s , S_s );   ▷ Set P_s as P_t's new server peer
24:         startDownload( P_t , S_s );
25:         startDownload( P_i , P_r , S_k );
26:         break;
27:       end if
28:     end for
29:   else                                  ▷ APAS Strategy – Elimination
30:     doAPAS_Elimination( S_k , H(S_k) );
31:                                         ▷ Refer to Algorithm 3
32:   end if
33: end procedure
```

Algorithm 3: Advanced Peer Assignment Strategy with Peer Elimination

```
1:  Procedure DoAPAS_Elimination( S_k , H(S_k) )
2:    H(S_k) ← sortContribution( H(S_k) , ↓ );
3:    P_low ← getLastPeer( H(S_k) );
4:    if P_i's C_i > P_low's C(low) then
```

```
5:      P_r ← findServerPeer( P_low , S_k );
6:      stopDownload( P_low , S_k );
7:      startDownload( P_i , P_r , S_k );
8:      else                                          ▷ P_r = null
9:          Unsuccessfully! P_i's request for S_k;
10:     end if
11: end procedure
```

4.5 Distributed Peer Assignment

Further modification allows distributed peer assignment strategies to be used in large-scale distributed P2P networks. The APAS and APAS-E with uncomplicated modifications are still compatible with distributed P2P systems. This subsection describes large-scale distributed P2P network environments and designs two tracker-oriented and peer-oriented approaches for distributed APAS, called D-APAS for brevity.

4.5.1 Partially-Centralized and Structured P2P Environment

The practical deployment of a large-scale P2P system is usually based on the *partially-centralized* and *structured* P2P model, such as the P2P overlay categories in [3]. Such a P2P system consists of a number of smal-scale centralized and structured P2P systems, which accord with the conventional system described in Section 3.1. Every small P2P system is referred as a domain D_i, and contains a hosting tracker T_i in charge of peers that exist in this domain. Trackers from external P2P domains form and communicate in a tracker group TG, called a "closed tracker network." In this network each tracker acts as a local cental directory for peers in its local domain. 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 over the tracker network. Although an unstructured P2P system is somewhat workable in a small-scale network, combining many unstructured P2P systems into a large one is inpracticable due the drastic increase of message workload and tremendous search requests flooding the system. Instead, the proposed D-APAS is based on a partially-centralized and structured network architecture.

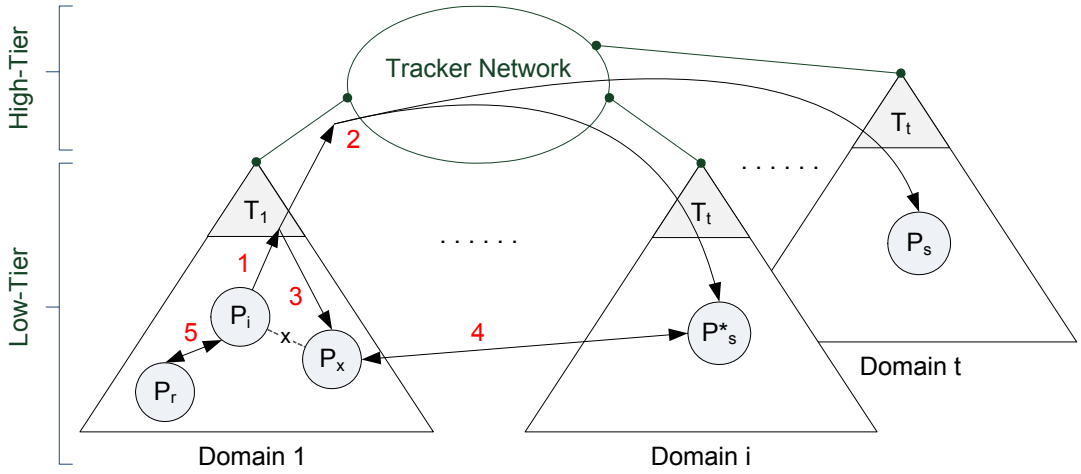


Fig. 6. Distributed peer assignment strategies in a large-scale P2P network.

Fig. 6 illustrates this reference architecture, which introduces a new tracker network. A tracker group consists of many trackers that enforce PAS, APAS, and APAS-E in their P2P domains. Trackers communicate with other trackers to collabrate on D-APAS over the tracker network. Due to a varity 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. This type of network commonly appears in many BitTorrent-like P2P applications. Consequently, the P2P netwok archiecture can be regarded as an instance of a two-tier system archtecture.

4.5.2 Distributed APAS (D-APAS)

The D-APAS attempts to augment the effects of APAS and APAS-E, as described in Sections 4.3 and 4.4, in a large-scale P2P network environment. D-APAS 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 basic PAS procedure within a local domain remains unchanged. What is mainly modified in D-APAS is the procedure of generating a set of candidate server peers, i.e., $H^*(B(S_k))$, from a tracker group instead of a singular tracker, as compared in Step 2 in Section 4.3.3. In addition, D-APAS is based on several assumptions.

- 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.

In reference to **Fig. 6**, the following steps describe the designs of D-APAS and its procedure (Steps 1-5). To ease presentation, Algorithm 4 shows the algorithmic form of the D-APAS. Initially, D-APAS will start in case that a peer P_i with request r^i_k 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. (This step is similar to Step 1 in Section 4.3.3)

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 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. Remember that every tracker follows the same process in Step 2 in Section 4.3.3 to find a potential substitute within its domain.

To produce P^*_s, the development of D-APAS presents the tracker-oriented and the peer-oriented methods, respectively described as follows.

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

  Firstly, $T_i(S_k)$ 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.

Next, 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 in D_x. Thus, T_x reports P_s to $T_i(S_k)$.

Finally, since all external trackers perform the above process simultaneously, $T_i(S_k)$ can receive a numer of reports from external trackers. Accordingly, $T_i(S_k)$ figures out the optimal substitute P^*_s.

- 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.

  Firstly, 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 <segment, tracker> = <S_x, T_x>. 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 $B(S_k)$.

  Next, $T_i(S_k)$ 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.

  Subsequently, T_x performs the same procedure as mentioned above to determine $H^*(B_{sub}(S_k))$ and a potential substitute P_s 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 examines 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 $H(S_k)$. Since this step is similar to the procedure of Step 3 and Step 4 in Section 4.3.3, the details of this step are omitted here.

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.

Algorithm 4: Distributed Advanced Peer Assignment Strategy

```
1:  Procedure Do_D-APAS_OptimalSubstitute ( S_k, H(S_k), T_i(S_k,) )
2:    B(S_k) ← findSegmentSet( H(S_k) );
3:    if TrackerOrientedMethod is True then
4:      t ← getNumOfTrackers( TG );
5:      for T_t ∈ TG and t ← 1, T do
6:        P_s ← findPotentialSubstitute_CBF( H(S_k), B(S_k), T_t , D_t );
7:                                          ▷ CBF(): a call-back-function
8:                  ▷ This function is similar to Lines 1-14 in Algorithm 2
9:
10:       H*( TG ) ← H*( TG ) ∪ P_s ;
11:     end for
12:   else                                          ▷ PeerOrientedMethod
```

```
13:       n ← getNumOfPeers( H(S_k) );
14:       for P_n ∈ H(S_k) and n ← 1, N do
15:          L_<S,T> ← getListOfPotentialTracker( P_n );
16:          TG_sub ← TG_sub ∪ L_<S,T>;
17:       end for
18:       t ← getNumOfTrackers( TG_sub );
19:       for T_t ∈ TG_sub and t ← 1, T do
20:          P_s ← findPotentialSubstitute_CBF( H_sub(S_k), B_sub(S_k), T_t , D_t );
21:          H*( TG ) ← H*( TG ) ∪ P_s ;
22:       end for
23:    end if
24:    if H*( TG ) ≠ {∅} then
25:          H*( TG ) ← sortGrade( H*( TG ) , ↓ );
26:          P*_s ← getFristPeer( H*( TG ) );      ▷ P*_s: an optimal substitute
27:          ……
28:          /* The omitted code is the same as Lines 15-28 in Algorithm 2 */
29:          ……
30:    Else
31:       doASAP_Elimination( S_k, H(S_k) );
32:                                                  ▷ Refer to Algorithm 3
33:    end if
34: end procedure
```

4.6 Bandwidth Adjustment and Feedback

Peer assignment inevitably addresses the potential issue of asymmetric bandwidth allocation in a dynamic P2P context. According to the *flat* bandwidth allocation above, the hosting server/tracker can estimate the upload bandwidth that a server peer assigns to a client peer based on the server peer's grade. In practice, a client peer can have more, equal, or lower unoccupied download bandwidth than its server peer's upload bandwidth. Thus, asymmetric bandwidth allocation can cause lower resource utilization and longer transfer time. This problem calls for a dynamic bandwidth adjustment method capable of tackling the different asymmetric scenarios caused by variances in request workload, access pattern, peer churn, etc. Ideally, the server should be able to reclaim unused and surplus bandwidth resources, and reallocate them based on the relative contributions of client peers. Client peers with higher contributions can obtain more bandwidth resources, achieving efficiency and fairness in bandwidth utilization.

The following discussion explains the dynamic bandwidth adjustment method. This design controls several factors, including segment access popularity, segment replication and scarcity, request workload, and service utilization based on the amount of pending requests, in a dynamic P2P context. In the spirit of the P2P paradigm and from a causality standpoint, distributing the rare/scarce segments requested by many pending requests should be considered first to avoid dropping requests. Client peers can commonly request a particular segment, but often get stuck in sleep and idle states, even though the aforementioned substitute policy is used. This phenomenon indicates that the overall supply of upload bandwidth for this segment falls short of demand, and implies an insufficient amount of server peers. In this case, the server should find new or more peers to contribute to this segment. Accordingly, the following measure calculates the relative "immediacy" of any segment S_k in the system. The segment of the largest value of $I(S_k)$ should be uploaded first.

$$I(S_k)=\frac{\left|sleep(S_k)\right|+\left|idle(S_k)\right|}{\left|H(S_k)\right|}, \tag{3}$$

where the term $|H(S_k)|$ indicates the cardinality of $H(S_k)$, i.e., the number of peers having S_k in the system, and $|sleep(S_k)|$ and $|idle(S_k)|$ represent the number of peers asking for S_k in the sleep and idle states, respectively.

This study learns that the most cost-effective course of action is to apply bandwidth adjustment to peer's joining and leaving processes, resulting in a low-complexity system design. Assume that a peer is free to join and leave a P2P system anytime. Initially, a new peer without contribution is not comparable to other peers in the system. With service provision, the server must help new peers offer their upload capacities as soon as possible to accumulate contributions. Given a set of segments that a new peer owns, the server checks with its tracker to match and direct some client peers' downloadings to this new peer. The tracker then arranges a group of target client peers for a new peer. Peers in the idle state are checked before peers in the sleep state. Other peers in the active state will be skipped, while they are presently satisfied by other server peers. Hence, this checking order reduces the query drop rate caused by idle peers.

Finally, consider the case of a server peer or a client peer leaving the network. In the former, all client peers downloading segments from a leaving server peer will be asked to re-send their requests. The hosting server then performs the necessary specific peer assignment and bandwidth adjustment to re-arrange download requests. In the latter case, the hosting server releases all upload bandwidth occupied by this client peer immediately upon detecting its departure. The returned upload bandwidth can be used to upload more new segments, or distributed to other ongoing uploading sessions.

5. Performance Evaluation

This section describes the simulation and evaluation of PAS, APAS, and APAS-E in terms of average download time, ratio of pending requests, and amount of download segments under different bandwidth capacity and peer contribution thresholds. This section also examines the relative performance of the tracker-oriented D-APAS and the peer-oriented D-APAS in terms of messaging overhead created by every request across the network group in the distributed network environment.

5.1 Simulation Environment

The experiments in this study involve a simple simulator based on a discrete and slotted time-based model. **Table 2** lists the parameters and parameter values used in the simulations. The simulator runs two different system models: 1) a centralized P2P model with a small-scale peer population of n=2048, 4096, or 8192 peers, and 2) a distributed model containing n=100000 peers evenly scattered in TG=1, 2, …, 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 μ_i=100 and σ=1 or a random distribution with μ_i=[50, 150]. The minimal threshold of remaining upload/download bandwidth is set as μ, that is a percent of μ_i.

Table 2. Description of simulation parameters.

Symbol	Meaning	Value range
t	time duration in the simulation	1000 ~ 8000
n	amount of peer population in a centralized model	2048, 4096, 8192
	amount of peer population in a distributed model	100000
TG	amount of trackers in the tracker network	2, …128
μ_i	a peer P_i's upload / download bandwidth capacity	Normal(100, 1) or Random(50, 150)
μ	minimal / base unit of data transfer rate (percent of μ_i)	10, 20,…, 100%
d	amount of files in the system	820
f	file size	30,000 KB
m	uniform segment size	3,000 KB
α	tunable parameter to adjust relative weighting between UL_{past} and UL_{now}	[0, 1]
p_{in}	random prabability that a new peer joins in the system	0.002
p_{out}	random probability that a peer leaves as it has any pending request (or not)	0.002 (or 0.003)
p_r	random probability that a peer sends a new request	0.005 ~ 0.1
Q	range of mean relative duration	30

The simulation runs in 1000, 2000, 4000, or 8000 time units, and peers are free to join and leave the system. Particularly, 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, 0.01, 0.05, or 0.1. 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. In addition, a peer's contribution is increased by Eq. (1), in response to its segment uploading, as α is set in the range of 0.5 ± 0.15.

Accordingly, this study examines the proposed PAS, APAS, APAS-E, and D-APAS in terms of the four performance metrics below. The first two measures examine service agility and throughput in PAS, APAS, and APAS-E under variance of μ threshold. The third measure examines the effect of APAS on service fairness against free-riding issues. The last measure reflects the efficiency of D-APAS by considering messaging overhead in a large-scale distributed P2P network.

- *Average download time* means the average duration from the moment at which a client peer sends a segment request until the segment is uploaded by a server peer.
- *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.
- *Average number of download segments by contribution range* is the amount of segments that all peers in a specific contribution range have downloaded to the total of peers in this contribution range.
- *Average number of outward messages per request* indicates the average of outward messages among trackers, caused by any request in a distributed P2P model.

5.2 Sensitivity to Average Download Time

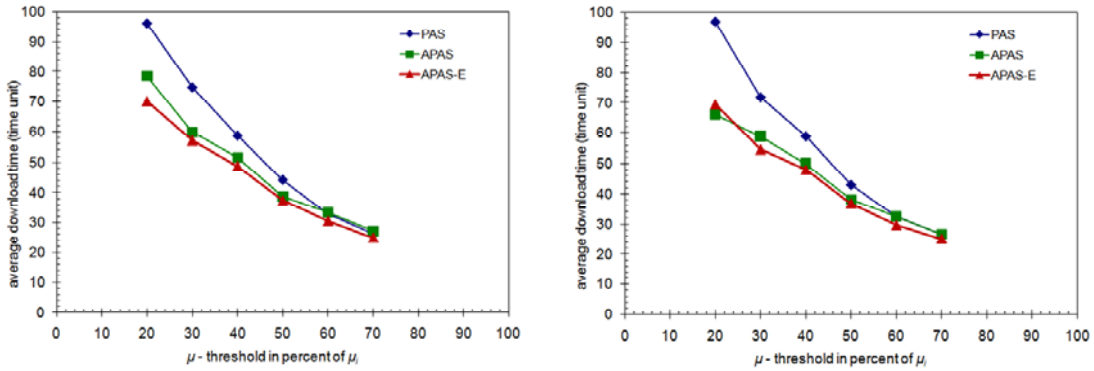
This subsection inspects the radical intention of improving service agility and fairness. **Fig. 6** and **7** illustrate the experimental results of average download time attained by PAS, APAS, and APAS-E. **Fig. 8** compares the performance of PAS and APAS when the simulation runs over different time periods. **Fig. 9-(a)** presents similar results under both normal and random

distributions of peer capacity μ_i. These figures indicate that bandwidth granularity directly influences average download time. A larger value of μ decreases the download time; in contrast, a smaller value of μ improves download concurrency and service agility. APAS achieves better performance than PAS, and APAS-E achieves performance similar to that of APAS.

An analysis of the performance results above shows that the granularity of upload/download bandwidth allocation significantly influences system performance. The time difference between PAS and the other two methods increases as μ decreases. Because a smaller μ means a higher bandwidth granularity, the number of segments which a peer can upload and download increases. In APAS with the substitute policy, the number of candidate server peers for each segment request increases implicitly. Thus, a client peer has more chances to get an appropriate server peer and quickly begin downloading. When μ>50%, APAS and PAS have similar performance. In this case, the substitute policy can run in vain because it is not easy to find a server peer with enough remaining bandwidth to serve a passed client peer.

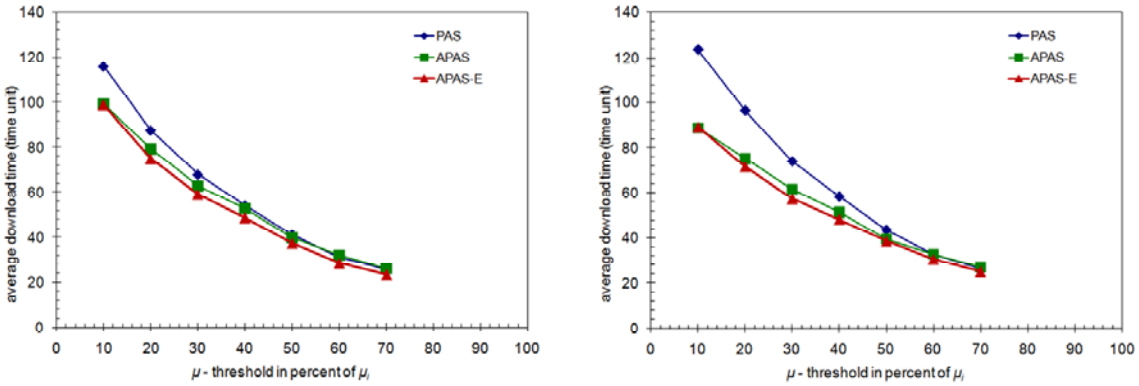
The efforts of APAS and APAS-E are prominent in reducing download time, especially under a heavy request workload as the factor of p_r is considered. **Fig. 6** shows that with $\mu \leq$ 50%, n=4096 or 8192, and p_r=0.1, APAS and APAS-E reduce download time by about 15~30% compared with PAS. In contrast, under a light request workload as shown in **Fig. 7-(a)** with n=4096 and p_r=0.005, APAS and APAS-E reduce download time by about 10~15% compared with the control result of about 12~30% in **Fig. 7-(b)** with p_r=0.05. This phenomenon also appears in **Fig. 8** with n=8192 and p_r=0.005, where all APAS's cases have less than 12 % of download time by comparing **Fig. 8-(b)** with **Fig. 8-(a)**.

APAS and APAS-E achieve similar performance regardless of μ since peer elimination is not meant to reduce the average download time. High bandwidth granularity can augment the effect of substitute policy in APAS and speed up request downloading with low blocking probability. In other words, when every peer can process more concurrent requests, the effect of elimination in APAS is minor relatively. On the other hand, the results above provide information about potential variations in block granularity of file segmentation subject to bandwidth granularity of data transfer capability, and vice versa, in a P2P file system.



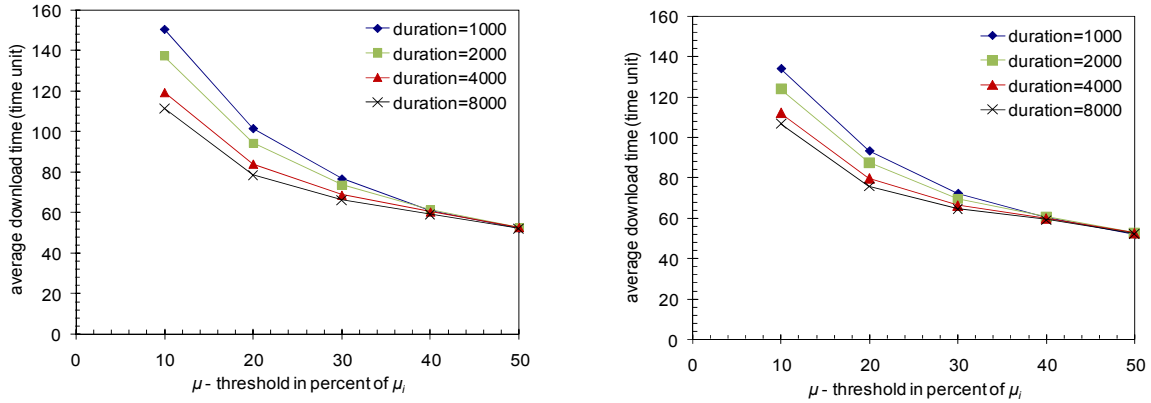
(a) peer population n=4096 (b) peer population n=8192

Fig. 6. Sensitivity to average download time under variance of μ (as p_r=0.1, t=1000 and α=0.5).



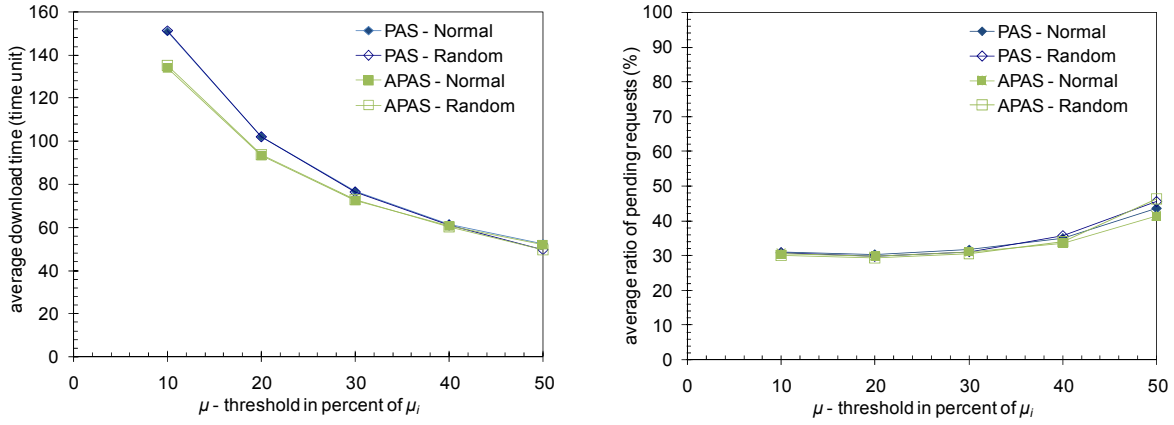
(a) p_r=0.005 (b) p_r=0.05

Fig. 7. Sensitivity to average download time under variance of μ (as n=4096, t=1000 and α=0.5).



(a) PAS (b) APAS

Fig. 8. Sensitivity to average download time under various durations (as n=8192, p_r=0.005 and α=0.5).



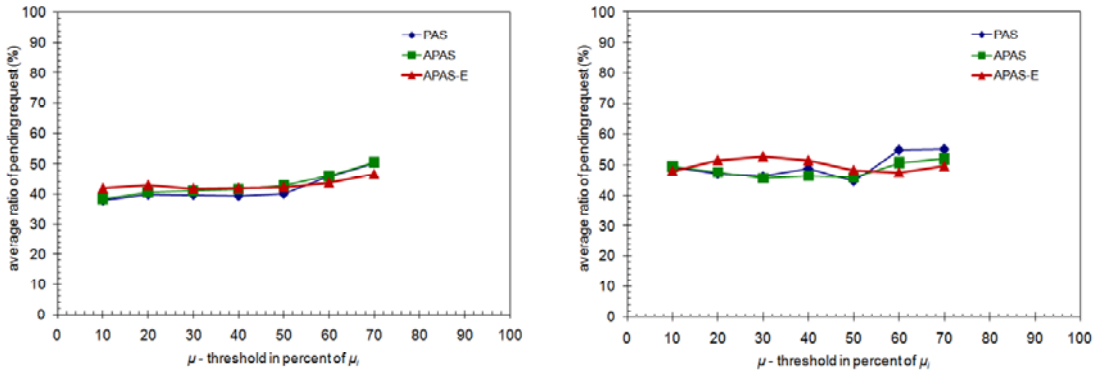
(a) average download time (b) average ratio of pending requests

Fig. 9. Performance sensitivities under variance of μ assigned by normal and random distributions (as n=8192, p_r=0.005, α=0.5, t=1000, and μ=Normal(100, 1) or Random(50, 150)).

5.3 Sensitivity to Average Ratio of Pending Requests

This subsection examines the influence of bandwidth granularity on the system throughput in terms of average ratio of pending requests. **Fig. 9** shows that all PAS and APAS curves under normal and random distributions of peer bandwidth capacity are very close. To avoid redudancy, Fig. 10 only depicts the results of normal distribution.

Note that all the results of PAS, APAS and APAS-E fluctuate only slightly, and are very similar. This is because system throughput, i.e., service capacity in a P2P system, is less sensitive to bandwidth granularity under stable request workload. As shown in **Fig. 9-(b)** and **Fig. 10**, although the average ratios obtained by PAS and APAS somewhat ascend as μ increases, its ascending degree is small yet. This is because a larger μ decreases the total number of available server peers for any segment request, but reduces the download time. However, a smaller μ enables a peer to have more concurrent segment requests, but it takes a longer time to transfer every segment. In addition, the increase of request workload *evenly* boosts the ratios of pending requests, for example, roughly from 40→50 as p_r=0.005→0.05 in **Fig. 10-(b)**. Hence, the system throughput exhibits no drastic variations. Compared with PAS, applying the substitute policy and peer elimination does not affect system throughput, thereby supporting the reliability of APAS and APAS-E.



(a) p_r=0.005 (b) p_r=0.05

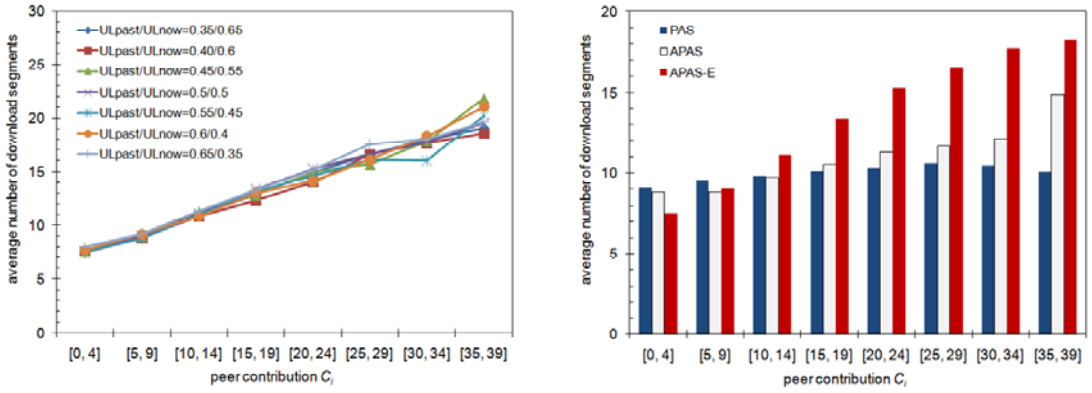
Fig. 10. Sensitivity to average ratio of pending requests (i.e, intensity of dropping requests) under variance of μ (as n=4096, t=1000 and α=0.5).

5.4 Sensitivity to Average Number of Download Segments

This subsection reviews various attempts to limit the free riding problem and maintain service fairness among peers with different levels of contribution. Specifically, Eq. (1) classifies peers into clusters based on their peer contributions. The performance results corresponding to each cluster are determined by different measures of average download segments, number of times by APAS-E, and request blocking ratio of APAS-E to PAS, as **Fig. 11-(b)**, **12-(a)**, and **12-(b)** respectively illustrate.

Fig. 11-(a) examines the sensitivity to relative weighting between a peer's past and current contributions in different peer contribution ranges. Based on the foregoing investigation, in which APAS-E attains the best service agility and throughput, **Fig. 11-(a)** only presents the results of APAS-E to simplify the presentation. All the curves in this figure go up almost linearly corresponding to incremental increases in the x-axis. Hence, simply applying Eqs. (1) and (2) to the peer assignment process helps penalize free riders. In addition, the variety of relative weightiness achieved by tuning α in Eq. (1) yields similar performance. Without loss

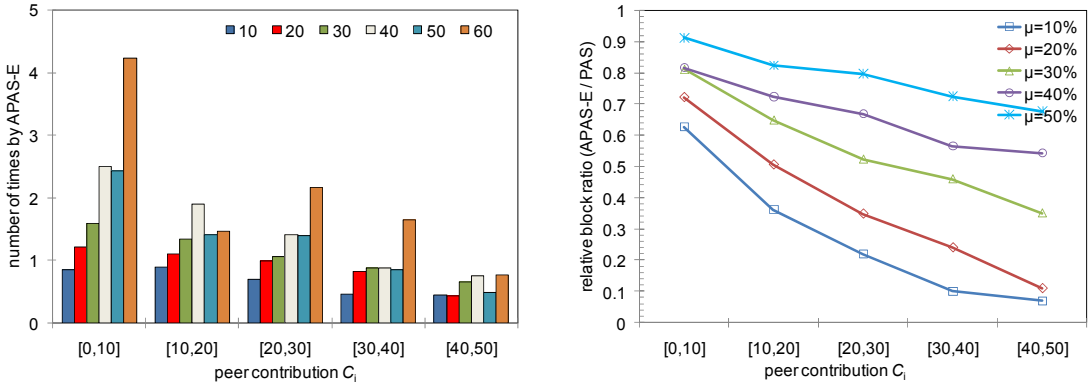
of generality, this study derives the experiment results of case $\alpha = 0.5$ to examine the relative performance of PAS, APAS, and APAS-E, as **Fig. 11-(b)** depicts.



(a) APAS-E: relative weighting by α

(b) relative performance

Fig. 11. Sensitivity to the average number of download segments with respect to different peer contribution ranges: (a) APAS-E to relative weighting between UL_{past} and UL_{now} (as n=8192, μ=50%, p_r=0.1, t=1000 and α=[0.35, 0.65]), and (b) relative performance among PAS, APAS, and APAS-E (as n=8192, μ=50%, p_r=0.1, t=1000 and α=0.5).



(a) APAS-E: occurrences under variance of μ

(b) relative performance

Fig. 12. Relative performance between APAS-E and PAS with respect to different peer contribution ranges: (a) number of times - the occurrences of peer elimination made by APAS-E (as n=8192, p_r=0.005, α=0.5, t=1000 and μ=10, …, 60%), and (b) block ratio of APAS-E to PAS (as n=8192, p_r=0.005, α=0.5, t=1000 and μ=10, …, 50%).

APAS-E outperforms the other methods in service fairness and utilization. A comparative baseline consisting of PAS with FIFO leads to a *flat* outcome when the difference between two peers in respective clusters is small, or less than 10% of the total number of download segments. Based on its tie-in substitute policy, APAS rewards frequently-contributing peers with more download resources. For instance, the number in cluster of [35, 39] is 1.5 times that in cluster [0, 4]. In addition to the substitute policy, the elimination method can significantly boost the performance, increasing the difference between these two clusters about 2.4 times. The results in **Fig. 12-(a)** suggest that APAS-E enforces fair peer elimination. In all cases, the number of peers eliminated from a cluster of lower contribution is larger under different

bandwidth granularities. Furthermore, **Fig. 12-(b)** shows the ratio of respective numbers of blocked requests in PAS and APAS-E caused by temporary service unavailability, during which no server peer or substitute server peer is available. APAS-E enables peers with high contributions to experinece the reward of high service utiliztion. This effort is quite conspicuous in the cases of high bandwidth granularity.

In summary, the previous investigations of a centralized P2P context indicate that APAS with the substitute and elimination policies is amenable and efficientdue to its beneficial effects on system throughput and service fairness. The remainder of this section examines its distributed version, i.e., D-APAS, in a large-scale distributed P2P environment.

5.5 Performance Results of D-APAS

This subsection shows that D-APAS is able to maintain service utilization and fairness in a distributed P2P environment. As Section 4.5 shows, this study simulates and compares tracker-oriented and peer-oriented D-APASs in terms of messaging overhead across the tracker group into the distributed network, where every tracker still performs APAS-E as usual inside its network domain. **Fig. 13** compares the performance of tracker-oriented and peer-oriented D-APASs in a large-scale context with n=100000. 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.

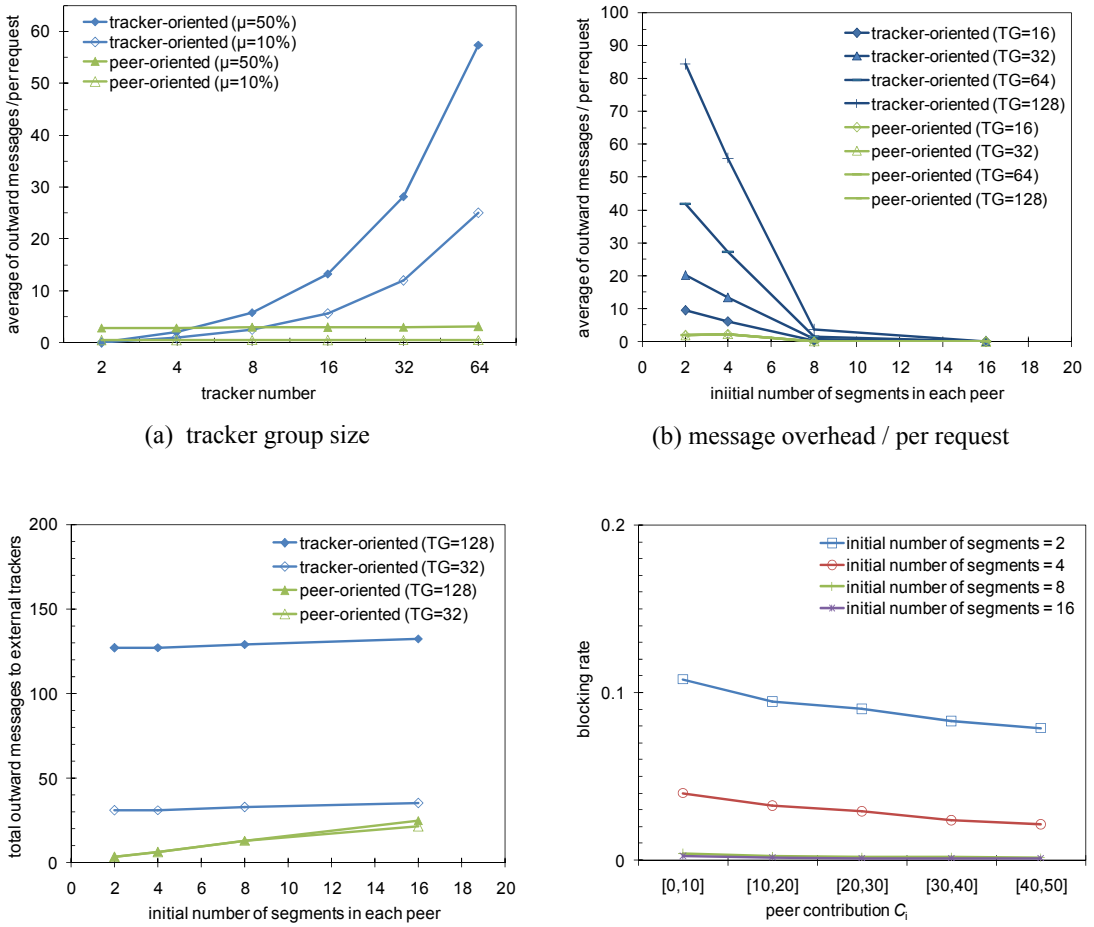
The peer-oriented method outperforms the tracker-oriented method in terms of outward message overhead. The experimental results in **Fig. 13(a)-(c)** show some observations below.

First, the tracker-oriented method induces a linear growth in message overhead as the tracker group size increases, as **Fig. 13-(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 because it efficiently reduces the overall message overhead. Although this pre-determining process generates some computation and communication overheads, these costs are limited within 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. 13-(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 μ=10% and μ=50% depicted in **Fig. 13-(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 using either PAS or APAS within the domain. On the other hand, the message overhead of the peer-oriented method remains low for the same reason as mentioned above.



(a) tracker group size

(b) message overhead / per request

(c) message overhead

(d) peer contribution

Fig. 13. Performance results by D-APAS (as n=100000, TG=[2, 4, …, 128], p_r=0.005, α=0.5, t=1000, and μ=[10, 30, 50%]): (a) average number of outward messages per request against tracker group size (as only one initial segment is set), (b) average number of outward messages per request against the initial number of segments in each peer (as μ=30%), (c) total of outward messages across the tracker network against the number of initial segments in each peer (as μ=30% and TG=128), and (d) block ratio in a peer cluster against the number of initial segments in a peer (as μ=30% and TG=128).

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. **Fig. 13-(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 seriously reduced when the system has segments scattered over the distributed network.

The discussion above reviews the peer-oriented and tracker-oriented methods of performing APAS and APAS-E in a distributed P2P network environment. Based on the relative performance indicated in **Fig. 13**, 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. Therefore, though the tracker-oriented method is not economical, it is a viable alternative under certain conditions.

6. Conclusions

This study addresses the peer churn and free riding problems appearing in P2P networks. To mitigate these problems, this study proposes a simple and efficient peer assignment scheme to protect against performance degradation in terms of service capacity and fairness. The proposed design enforces prioritized admission and scheduling policy and deals with request workload and upload/download resource allocation in a fair and efficient manner. This study also examines several implicit issues, including download concurrency; download blocking, and service starvation. This study designs supplementary methods, including peer substitute, peer elimination, and bandwidth adjustment, to resolve these issues and prevent APAS performance degradation. Experimental results confirm that APAS enhances bandwidth utilization and achieves fair resource allocation for P2P file sharing applications.

This study extends the design of APAS to create the D-APAS, which performs well in large-scale distributed P2P networks. This study presents and compares two variants of peer-oriented and tracker-oriented D-APASs. Performance results show that both methods maintain the functionalities of APAS and augment the effects of peer substitute and elimination in distributed P2P networks. 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 contributions of this study are two-fold: the proposed APAS and D-APAS can be used in centralized and distributed P2P network environments, respectively, to ensure fair peer assignment strategies for P2P file sharing applications. Extensive simulations confirm the efficacy, applicability, and extensibility of these approaches.

References

[1] The cooperative association for Internet data analysis (CAIDA), “Internet traffic classification,” online available: http://www.caida.org/research/traffic-analysis/classification-overview/, 2009.

[2] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, “Modeling peer-peer file sharing systems,” in *Proc. of IEEE INFOCOM’03*, vol. 3, pp. 2188-2198, 2003.

[3] 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, 2004.

[4] X. Yang and G. de Veciana, “Service capacity of peer to peer networks,” in *Proc. of IEEE INFOCOM’04*, vol. 4, pp. 2242-2252, 2004.

[5] J. S. Kong, J. S. A. Bridgewater, and V. P. Roychowdhury, “Resilience of structured p2p systems under churn: the reachable component method,” *Computer Communications*, vol. 31, no. 10, pp. 2109-2123, 2008.

[6] D. Stutzbach, and R. Rejaie, “Understanding churn in peer-to-peer networks,” in *Proc. of the 6th ACM SIGCOMM Conf. on Internet measurement*, pp. 189-202, 2006.

[7] 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, 2005.

[8] C.-L. Hu and T.-H. Kuo, “Hierarchical peer-to-peer overlay with cluster-reputation-based

adaptation," in *Proc. of the 2009 IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, 2009.
[9] F. E. Bustamante and Y. Qiao, "Designing less-structured p2p systems for the expected high churn," *IEEE/ACM Trans. on Networking*, vol. 16, no. 3, pp. 617-627, 2008.
[10] E. Adar and B. A. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, no. 10, 2000.
[11] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. of the SPIE Vol.4673: Multimedia Computing and Networking*, 2002.
[12] 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, 2008.
[13] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *ACM SIGecom Exchanges*, vol. 5, no. 4, pp. 41-50, 2005.
[14] P. Antoniadis, C. Courcoubetis, and R. Mason, "Comparing economic incentives in peer-to-peer networks," *Computer Networks*, vol. 46, no. 1, pp. 133-146, 2004.
[15] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, no. 12, pp. 45-48, 2000.
[16] S. Jun and M. Ahamad, "Incentives in BitTorrent induce free riding," in *Proc. of the 2005 ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems*, pp. 116-121, 2005.
[17] D. Hughes, G. Coulson, and J. Walkerdine, "Free riding on gnutella revisited: the bell tolls?" *IEEE Distributed Systems Online*, vol. 6, no. 3, 2005.
[18] B. Yang and H. Garcia-Molina, "PPay: Micropayments for peer-to-peer systems," in *Proc. of the 10th ACM Conf. on Computer and Communications Security*, pp. 300-310, 2003.
[19] M. J. Osborne, An introduction to game theory, Oxford University Press, 2004.
[20] K. Eger and U. Killat, "Bandwidth trading in BitTorrent-like p2p networks for content distribution," *Computer Communications*, vol. 31, no. 2, pp. 201-211, 2008.
[21] I. Osipkov, P. Wang, and Y. Kim, "Robust accounting in decentralized p2p storage systems," in *Proc. of the 26th IEEE International Conf. on Distributed Computing Systems*, pp. 300-310, 2006.
[22] I. Simplot-Ryl, I. Traore, and P. Everaere, "Distributed architectures for electronic cash schemes: a survey," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 24, no. 3, pp. 243-271, 2009.
[23] E. Fourquet, K. Larson, and W. Cowan, "A reputation mechanism for layered communities," *ACM SIGecom Exchanges*, vol. 6, no. 1, pp. 11-22, 2006.
[24] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.
[25] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *Proc. of the 6th ACM SIGCOMM Conf. on Internet measurement*, pp. 203-216, 2006.
[26] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. of ACM SIGCOMM'04*, pp. 367-378, 2004.
[27] M. Li, J. Yu, and J. Wu, "Free-riding on BitTorrent-like peer-to-peer file sharing systems: modeling analysis and improvement," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 954-966, 2008.
[28] M. Yang, Z. Zhang, X. Li, and Y. Dai, "An empirical study of free-riding behavior in the maze p2p file-sharing system," in *Proc. of the 4th Annual International Workshop on Peer-To-Peer Systems*, vol. 3640 of Lecture Notes in Computer Science, Springer, pp. 182-192, 2005.
[29] A. R. Bharambe, C. Herley, and V. Padmanabhan, "Analyzing and improving a BitTorrent network's performance mechanisms," in *Proc. of IEEE INFOCOM'06*, 2006.
[30] H. Liu and C. Hsu, "Anne: a fair service capacity management for p2p overlay networks," in *Proc. of the 2nd International Conf. on Communications and Networking in China*, pp. 265-269, 2007.
[31] R. Jurca and B. Faltings, "Reputation-based pricing of p2p services," in *Proc. of the 2005 ACM SIGCOMM Workshop on Economics of peer-to-peer systems*, pp. 144-149, 2005.
[32] R. Prasad, V. Srinivas, V. Kumari, and K. Raju, "An effective calculation of reputation in p2p networks," *Journal of Networks*, vol. 4, no. 5, pp. 332-342, 2009.
[33] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc. of the 16th ACM International Conf. on Supercomputing*, pp.

84.95, 2002.
[34] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in *Proc. of the 22nd International Conf. on Distributed Computing Systems*, pp. 5-14, 2002.
[35] M. Yanga and Z. Fei, "A novel approach to improving search efficiency in unstructured peer-to-peer networks," *Journal of Parallel and Distributed Computing*, vol. 69, no. 11, pp. 877-884, 2009.
[36] C. Gkantsidis, M. Mihail, and A. Sabei, "Hybrid search schemes for unstructured peer-to-peer networks," in *Proc. of IEEE INFOCOM'05*, vol. 3, pp. 1526-1537, 2005.
[37] X. Shi, J. Han, Y. Liu, and L. M. Ni, "Popularity adaptive search in hybrid p2p systems," *Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 125-124, 2009.
[38] T. Lin, P. Lin, H. Wang, and C. Chen, "Dynamic search algorithm in unstructured peer-to-peer networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, no. 5, pp. 654-666, 2009.

Chih-Lin Hu received the BS degree in computer science from the National Cheng-Chi University in 1997, the MS degree in computer science from the National Chung-Hsing University in 1999, and the PhD degree in electrical engineering from the National Taiwan University in 2003. He was a researcher at BenQ and Qisda Advanced Technology Centers, Taipei City, Taiwan, from 2003 to 2007. Since 2008, he has been an assistant professor in the Department of Communication Engineering, National Central University, Taoyuan, Taiwan, R.O.C. He had the honor to get the best paper award in IEEE ICPADS 2000 and BenQ Innovation Awards in 2006 and 2007. He had co-organized MDM'09 Workshop on Mobile Peer-to-Peer Information Services (MP2PIS) and IEEE PerCom'10 Workshop on Mobile Peer-to-Peer Computing (MP2P). His research interests include mobile and pervasive computing systems, broadcast information system, digital home network, and Internet technology. Dr. Hu is a member of the ACM and the IEEE.

Da-You Chen received the BS degree in communication engineering from the National Central University, Taiwan, in 2010. He is currently pursuing the MS degree at the same faculty. His research interests include peer-to-peer system, delay tolerant network, and Internet technology.

Yi-Hsun Chang received the BS degree in electrical engineering from National Chi-Nan University, Taiwan, in 2008, and the MS degree in communication engineering from National Central University, Taiwan, in 2010. Her research interests include P2P networking and computing technologies. She is an IEEE student member.

Yu-Wen Chen was born in Taiwan, in 1988. She received the BS degree in communication engineering from the National Central University (NCU), Taiwan, in 2010. She is currently pursuing the MS degree at Columbia University, NY, USA. She had the honor to get the 2009 NCU Best Student Award. Her research interests include mobile and wireless communication systems, mobile computing, and Internet technology. She is an IEEE student member.