

Simple Peer Selection Strategies for Fast and Fair Peer-to-Peer File Sharing

Chih-Lin Hu, Yi-Hsun Chang, Da-You Chen and Yu-Wen Chen

Department of Communication Engineering, National Central University, Taoyuan, Taiwan, R.O.C.

clhu@ce.ncu.edu.tw; {975203043, 955003009, 955003003}@cc.ncu.edu.tw

Abstract—This paper proposes an adaptive peer selection strategy in support of fast and fair peer-to-peer file sharing applications. According to specific features of peer duality, the system server is able to schedule download requests by degrees of peers' contributions, and correspondingly assign server peers in charge to process requests by degrees of peers' capacities. In addition, several functional extensions, substitute and elimination processes, are further associated to cope with subtle situations of service starvation and download blocking, and hence make the system design robust and amenable. Simulation results exhibit that the proposed mechanisms are simple, but effective in maintenance of service agility, fairness and resource utilization, affected by critical peer churn and free riding problems.

I. INTRODUCTION

The peer-to-peer (P2P) networking has recently emerged as a new variant of distributed computing paradigm for building distributed networked applications. P2P applications generate dominant fraction of today's Internet traffic that is chiefly associated with P2P file sharing applications [1]. In P2P file sharing applications [2], like Napster, Gnutella, KaZaA, eDonkey and BitTorrent, a peer both requests files from its peers, and stores and serves files to its peers. An increase of peer population results not just in an increase of workload, but also in an increase of service capacity to process the workload requests [3]. Hence, *peer duality* makes the P2P approach distinguishable from the traditional client/server approach where clients and servers are distinct so that an increase of client population simply leads to an increase of workload, thereof will the scalability problem and performance degradation arise.

Dynamics of peer participation, *peer churn*, influences overlay design, resiliency and assessment [4]. Peers in a system cooperate to run an application-level overlay network [5] that provides connectivity, signal messaging, routing, discovery and searching between end hosts that are addressable on top of IP networks. However, due in part to peers' autonomy and mutual dependency, the transiency of peers and its implications have a great impact on the efficacy of replication, search and query mechanism in content distribution [6]

The problem of *free riding* in a P2P system goes against reciprocal provision of resources among peers. As examined [7], a large fraction of peer population is deemed *free riders* – maximizing their own utilities that are out of proportion to their contributions to the system. Free riding is irrelevant to P2P networking, but inherits from P2P participants' attitudes of mind. To encourage peer contribution and keep service

fairness, various *incentive* approaches are used to move peers to do contributions [8][9][10]. Among others, the growing recognition of reputation systems has recently led to a significant research dimension. Reputation-based schemes take the histories of peers' contributions to the system as their decision making processes [11]. Reputable peers should be served in high priority and rewarded more resources to compensate for their offering. Our intention then follows this mainstream as the development of resource allocation and distribution mechanisms in P2P file sharing contexts.

By considering dual factors of client peer contribution and server peer capacity, this paper designs a simple peer selection strategy for fast and fair P2P file sharing. This proposes to maintain service agility and fairness among peers against peer churn and free riding. Particularly, the basic idea is to gradate peer contribution and capacity. Given with request workload, the system server applies a specific peer selection strategy to control the admission and scheduling process to guarantee fair and efficient use of resources, yet without loss of service capacity and system throughput in a system. Therefore, the results of this paper work are summarized as follows. First, the primary work formulates the measures of peer contribution and server capability. In light of indirect-reciprocity notion [8], a basic peer selection strategy (BPSS) is designed to choose server peers to process download requests in descending order of peer contribution. Second, the adaptive peer selection strategy (APSS), extending BPSS with a specific substitute policy, can modify peer assignment in response to traffic dynamics due to peer churn and request workload. Third, the APSS is associated with specific peer elimination process to cope with possible service starvation and further boost the ability of service differentiation and fairness to a prominent extent. Finally, experimental simulations are conducted to assess performance sensitivities in terms of several performance metrics, average download time, ratio of pending requests, download count by contribution range. Consequently, this paper presents an adaptive peer selection strategy with several auxiliary functions, thus sustaining system throughput, service agility and fairness against service starvation and traffic dynamics in P2P networks.

The rest of this article is organized as follows. Section II mentions the P2P system model. Section III designs the BPSS, APSS and auxiliary functions. Section IV presents performance results. Conclusion is given in Section V.

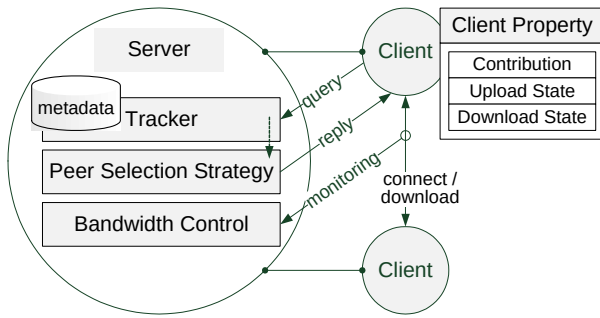


Fig. 1. A P2P file sharing context.

II. SYSTEM MODELING

This section describes a P2P system environment, and specifies bandwidth resource utilization and peer state management that are in use for resource allocation and content distribution.

A. System Environment

Fig. 1 shows a centralized, structured P2P system environment [2] where the proposal is designed. Specifically, there is a central server that functions tracking and provision services in support of distributing file segments inside a P2P network. A tracker server is located in a central server or another dedicated host adhered to a central server. It maintains segment meta-data repository, segment management, and peer management in the system. Functionally, a tracker 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 classifies them by their download states. Thus, every peer connects to the central server and inquires about where any indicated segment can be downloaded. The server then selects one out of many server peers according to a specific peer selection strategy. With a request reply, a client peer directly connects to its assigned server peer to access the indicated segment. During downloading, the central server can monitor the download session and timely adjust bandwidth allocation. In addition, the server peer can record download state of every requested segment, remaining upload bandwidth and uploading contribution into its property profile.

B. Resource Management

The P2P system enforces a working premise: the granularity of upload/download bandwidth allocation is based on a slotted time model; it takes one time slot to deliver each file segment. Each time slot has a minimal transfer rate μ as a base unit of bandwidth allocation. Accordingly, a file is divided into a number of segments of equal size.

For simplicity, let a peer P_i have uniform upload and download bandwidth capacities with the maximal data transfer rate μ_i respectively. P_i can at most offer $\lfloor \mu_i / \mu \rfloor$ transfer sessions simultaneously in either download or upload way. Peers that have remaining transfer rate lower than μ will not be considered requesting/offering any segments until they reclaim

enough bandwidth to this end. This precondition is so applied to facilitate the analysis of bandwidth granularity.

Selecting a *fit* server peer is a decisive phase before a client peer starts to download any segment. The system does peer management by degree of bandwidth utilization. Specifically, the update and download states of a peer are defined below.

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 when its remaining upload bandwidth is less than μ . A normal peer is uploading segments to other peers and has remaining upload bandwidth higher than μ . A leisure peer has upload bandwidth higher than μ , but does not upload any segment right now.

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

III. PEER SELECTION STRATEGY AND MANAGEMENT

This section proposes the peer selection strategies. Section III.A formulates the measures of client peer's contribution and server peer's grading of bandwidth capacity, both used to differentiate request priority and assignment among peers. Section III.B designs a basic peer selection strategy (BPSS). Section III.C designs an adaptive peer selection strategy (APSS) that extends the BPSS with substitute policy on peer assignment. Section III.D further integrates APSS with the bandwidth reclamation method (APSS-E) to guarantee fair bandwidth allocation. Finally, the above presents a joint design of peer selection and management mechanism against peer churn and free riding in a P2P system.

A. Peer Contribution and Grading

To tackle the free riding problem, the essence of peer management adopts an incentive-based approach to encourage peers to contribute their own resources so as to get back at better performance. Notice that the widely deployed BitTorrent employs a direct-reciprocity and tit-for-tat incentive to encourage cooperative behavior between a set of peers performing coordinated exchange of files [12]. However, BitTorrent has inherent choking issues, though it can be moderated by the associated optimistic unchoking method, thus striking a trade-off [13]. In contrast, this scheme adopts an indirect-reciprocity-based incentive approach by referring to historical and accumulated contributions of upload bandwidth that a peer has voluntarily provided. Without loss of generality, the measure of peer contribution is given by a liner formula for the quantity that a peer has donated till now.

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

where UL_{now} is the numbers of segments a peer is uploading now, UL_{past} is the numbers of segments which a peer has uploaded in the past, and $0 \leq \alpha \leq 1$ is a tunable parameter of

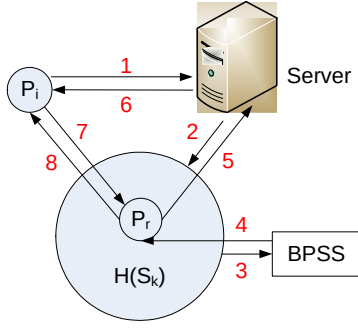


Fig. 2. A basic peer selection strategy.

relative weighting between two terms. Accordingly, the central server prioritizes download requests from client peers.

In processing upload bandwidth allocation, a central server decides which server peer to serve a request by considering the grading of upload bandwidth capacity that a server peer can provide to its client peers, as given by

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

B. Basic Peer Selection Strategy (BPSS)

The BPSS considers both service fairness and agility in the course of peer selection. Basically, BPSS appoints server peers by peer grading to take request workload scheduled by peer contribution. It 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 firstly. Then, a server peer of the highest G_i , from a number of candidates having the target segment, takes this request. Thus, BPSS can simply and evenly balance the use of upload bandwidth capacities among peers and to avoid bandwidth fragmentation scattering over the system.

Particularly, as depicted in Fig. 2, given with a peer P_i whose download bandwidth is higher than μ , P_i sends the central server a request r_k^i to download a segment S_k . With a r_k^i , the tracker generates a peer set, called $H(S_k)$, which includes all peers having S_k^i in their local storages. Then, the server selects a server peer P_r of the highest upload rate in $H(S_k)$. 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 will negotiate with P_r for downloading S_k . Else, the server will mark P_i 's download state as sleep and push this request back into the request queue. The server will later check r_k^i again if r_k^i is valid yet.

C. Adaptive Peer Selection Strategy (APSS)

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

1) *Concurrency*: Given with 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 $R(i)$. Like the way in BPSS, for each S_k in $R(i)$, the server quickly finds out a server peer from $H(S_k)$ for uploading S_k . If $R(i)$ contains more segments, this process will repeat till P_i 's remaining download bandwidth is less than μ . Herein, the processing sequence of requests in $R(i)$ however depends on any specific schedule, such as first-in-first-out (FIFO), earlier deadline first (EDF) and high download speed first, etc. Note that the study of request scheduling methods is orthogonal to this paper work that rather adopts the FIFO simply as a development baseline.

2) *Non-blocking*: Following BPSS, there is a critical situation of download blocking due to "service starvation." This situation – no uploading of any segment S_k in $R(i)$ – does not simply mean that a requested segment is inexistent in the system. For example, heavy request workload, asking for scarce segments, peer churn, and skewed access pattern can induce this situation, while *traffic dynamics* is considered. When all peers owning S_k are in busy state and have no more bandwidth to upload segments, all requests for $R(i)$ are blocked until any P_r in $H(S_k)$ reclaims sufficient upload bandwidth to take a new request. As examined, BPSS likely has in poor performance due to service starvation.

3) *Substitute Policy*: To resolve this situation, a substitute policy is specified to find a new substitute peer P_s to replace a busy peer P_r in $H(S_k)$ and to take over its ongoing uploading tasks. A replaced peer can then reclaim more upload bandwidth to take another request that asks for some segment now held by P_r . Accordingly, this policy can broadly alleviate the service starvation with respect to many influential factors under dynamic traffic. The following specifies the substitute policy and its procedure (Steps 1-4) in reference to Fig. 3.

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

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

Step 3: With a P_s , the central server tries to reclaim some upload bandwidth to resolve the blocking. P_s checks all its owned segments, denoted as $O(s)$, one by one. For every segment S_s in $O(s)$, if S_s is also included in $B(S_k)$, it means that at least one peer in $H(S_k)$ is currently uploading S_s to another peer in the system, too. So, the server proceeds to collect the set of client peers that are downloading S_s from peers in $H(S_k)$. Among these client peers, the one with the highest contribution¹ is chosen as a "passed" client peer.

Step 4: The server instructs a passed client peer to download S_s from P_s in place of its original server peer in $H(S_k)$. Then,

¹Explanatorily, a passed client peer may get better service after changing its server peer. If a P_s 's grade is less than its original server peer, the central server may alternatively check another P_s with lower C_s , but higher G_s .

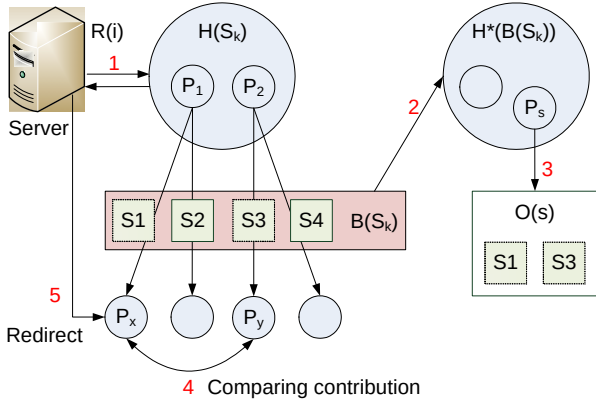


Fig. 3. The APSS with substitute policy.

the original one can get some uploading bandwidth back, and carry on uploading S_k to P_i .

Particularly, Fig. 3 instantiates the use of substitute policy. Let $H(S_k)$ include two peers, P_1 and P_2 . P_1 is busy in delivering S_1 to P_x and S_2 to another peer, and P_2 is busy in 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 $H^*(B(S_k))$ that includes all peers sending any segments in $B(S_k)$. Let the server select P_s in $H^*(B(S_k))$ as a candidate server peer. 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 decides P_x or P_y , the one with 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 serve one more pending request in queue.

D. APSS with Peer Elimination (APSS-E)

Following APSS, there is a subtle situation that a central server cannot find any suitable substitute server peer, since none of peers in $H^*(B(S_k))$ has enough remaining upload bandwidth. A peer elimination method is further devised to cope with the situation. Basically, this method attempts to pull back or sacrifice some peer's ongoing downloading if this peer has low contribution, like a free rider. Hence, a peer with higher contribution can preempt a downloading channel for fair resource allocation. Fig. 4 shows the procedure of APSS with peer elimination. By following the previous example in Fig. 3, suppose that the central server cannot find out a substitute server peer P_s to upload a particular $S_k \in R(i)$. The server checks all potential server peers in $H(S_k)$ and their client peers. Among these client peers, the one 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 victimizes P_{low} 's ongoing download, and can instruct P_{low} to resend a request for S_k , or alternatively defer its download till another server peer will become available later.

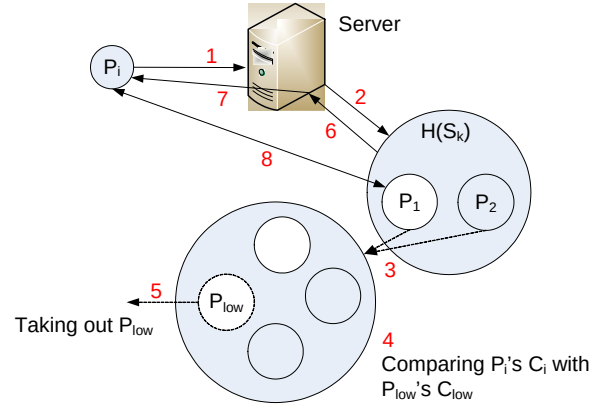


Fig. 4. The APSS with peer elimination.

IV. PERFORMANCE EVALUATION

This section describes the simulation and measurement, and evaluates BPSS, APSS and APSS-E in terms of average download time, ratio of pending requests, and amount of download segments under different thresholds of bandwidth capacity and degree of peer contribution.

A. Simulation Environment

The work in this paper has developed a simple simulator based on a discrete and slotted time-based model. The simulator initializes a P2P system that contains $n=8192$ peers. Every peer arbitrarily contains a file out of $d=820$ files as a file sharing base. Every file is of equal 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 using a normal distribution with $\bar{u}_i=100$ and $\sigma=1$. The minimal threshold of remaining upload/download bandwidth is u , as a percent of μ_i .

The simulation runs in 1000 time units; meanwhile, a peer is free to join and leave the system. Particularly, a new peer joins 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 $p_{out}=0.003$ at leisure/idle state. For traffic generation, heavy request workload is considered. Every peer issues a new request to access a file at a random probability $p_r=0.1$. Correspondingly, 10 segment requests are generated to perform concurrent uploadings/downloadings in a batch. Every peer is assigned a request queue of length $Q=30$ to keep its pending requests. In addition, a peer's contribution is increased by (1), in response to its uploading, as α is set in the range of 0.5 ± 0.15 .

Accordingly, the proposed schemes are examined in terms of three performance metrics below. Particularly, the first two measures are used to examine service agility and throughput among BPSS, APSS, and APSS-E under μ threshold variance. The last is to examine the effects of the APSS on service fairness against free-riding issues.

Average download time means the average duration from the moment at which a client peer sends a segment request till the

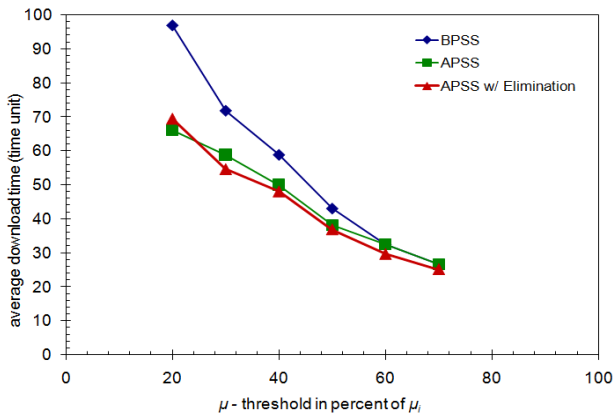


Fig. 5. Sensitivity to the average download time under variance of μ (baseline: $\alpha = 0.5$).

segment is uploaded by a server peer.

Average ratio of pending requests is the total 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 contribution range have downloaded to the amount of peers in this range.

B. Sensitivity to Average Download Time

This subsection inspects the radical intention of improving service agility and fairness. Fig. 5 illustrates the experimental results of average download time attained by BPSS, APSS and APSS-E. Obviously, APSS, an extension of BPSS with peer substitute policy, achieves better performance than BPSS. So does APSS-E that has similar performance to APSS's.

Scrutiny of performance results finds that granularity of upload/download bandwidth allocation has significant influence on the system performance. As shown, time difference between BPSS and other two increases as μ decreases. Explicitly, a smaller μ means a higher bandwidth granularity, so the number of segments which a peer can upload and download concurrently increases. When APSS with substitute policy is used, for any segment request, the number of candidate server peers increases implicitly. Thus, a client peer can have more chances to get an appropriate server peer and perform downloading in short time. But, as $\mu > 50\%$, APSS and BPSS have similar performance, while the substitute policy can run in vain because it is not easy to find a server peer of enough remaining bandwidth to serve a passed client peer.

Regardless of μ , APSS and APSS-E have similar performance since peer elimination is not meant to reduce the average download time. Therefore, high bandwidth granularity can augment the effect of substitute policy in APSS and speed up request downloading with low blocking probability. Namely, when every peer can process more concurrent requests, the effect of elimination in APSS is minor relatively.

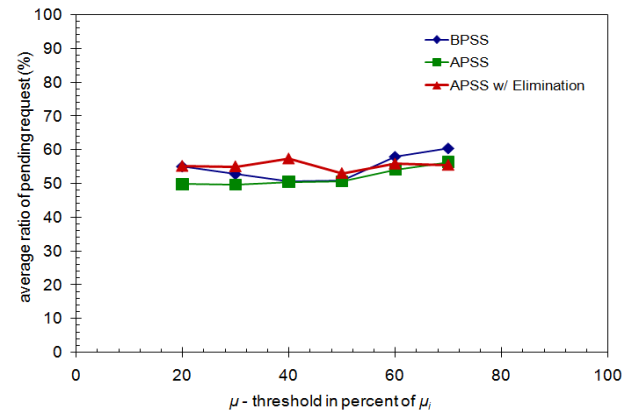


Fig. 6. Sensitivity to the average ratio of pending requests (i.e., intensity of dropping requests) under variance of μ (baseline: $\alpha = 0.5$).

C. 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. Observe in Fig. 6 that all results by BPSS, APSS and APSS-E are fluctuant slightly and very similar. It is explained that system throughput is less sensitive to bandwidth granularity under stable request workload. Although a larger μ can decrease the total of available server peers for any segment request, it however shortens the download time. Oppositely, a smaller μ enables a peer to have more concurrent segment requests, but it takes longer time to transfer every segment. Remarkably, as compared with BPSS, APSS does not affect system throughput after applying the substitute policy and peer elimination. This effect thereby supports the reliability of APSS and APSS-E.

D. Sensitivity to Average Number of Download Segments

This subsection inspects the efforts of restraining the free riding to maintain service fairness among peers with different contributions. Specifically, peers are classified into respective clusters according to their peer contributions given with (1). The average number of download segments in a cluster is figured out after the simulation. Fig. 7 are further derived to examine relative performance of BPSS, APSS and APSS-E.

APSS-E outperforms others in respect of service fairness and utilization. BPSS with FIFO, as a comparative baseline, leads to a *flat* outcome when the difference between two peers in respective clusters is small, about less than 10% of total number of download segments. Relatively, APSS makes a peer of higher contribution be rewarded more download resources, the event which is ascribed to the tie-in substitute policy. For instance, the number in cluster of [35,39] is 1.5 times that in cluster [0,4]. In addition to substitute policy, the elimination method is able to further boost the performance to a significant extent. Correspondingly, the difference between these two clusters is expanded to about 2.4 times. In summary, APSS with substitute policy as well as elimination is amenable and

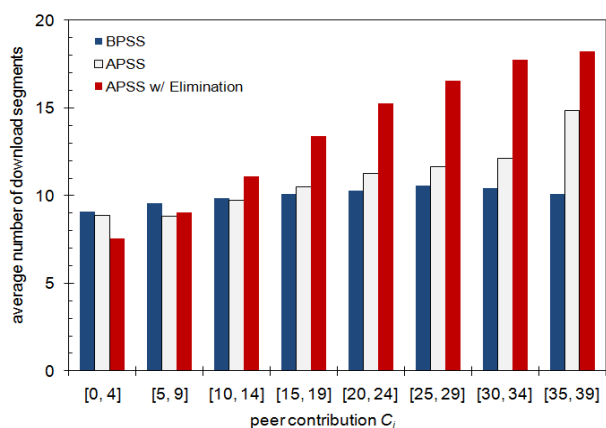


Fig. 7. Sensitivity of BPSS, APSS, and APSS with elimination to the average number of download segments with respect to different clusters of peer contribution (baseline: $\mu = 50\%$ and $\alpha = 0.5$).

efficient, as a result of profitable effects on system throughput and service fairness, after investigations in this section.

V. CONCLUSION

In this paper, we have clearly accounted for peer churn and free riding problems that are criticized in P2P networks. To avoid performance degradation of service capacity and fairness, we have proposed a peer selection strategy that performs adaptive peer management and assignment mechanisms. This design is so able to enforce prioritized admission and scheduling policy and deal with request workload and upload/download resource allocation in a fair and efficient manner. In addition, we have examined several implicit issues, download concurrency and blocking, and service starvation. Accordingly, supplemental substitute and elimination methods are designed to resolve these issues well. Experimental results have manifested that the functionality of the proposed APSS can not only enhance bandwidth utilization, but offer fair resource allocation in a dynamic context of P2P file sharing applications.

REFERENCES

- [1] CAIDA, The Cooperative Association for Internet Data Analysis, "Internet traffic classification," Online available: <http://www.caida.org/research/traffic-analysis/classification-overview/>, September 2009.
- [2] 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.
- [3] X. Yang and G. de Veciana, "Service capacity of peer to peer networks," in *Proceedings of IEEE INFOCOM'04*, vol. 4, pp. 2242–2252, March 2004.
- [4] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 189–202, October 2006.
- [5] 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.
- [6] 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.

- [7] E. Adar and B. A. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, no. 10, October 2000.
- [8] 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.
- [9] P. Antoniadis, C. Courcoubetis, and R. Mason, "Comparing economic incentives in peer-to-peer networks," *Computer Networks*, vol. 46, no. 1, pp. 133–146, September 2004.
- [10] K. Eger and U. Killat, "Bandwidth trading in bittorrent-like p2p networks for content distribution," *Computer Communications*, vol. 31, no. 2, pp. 201–211, February 2008.
- [11] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Communications of the ACM*, vol. 43, issue. 12, pp. 45–48, December 2000.
- [12] B. Cohen, "Incentives build robustness in bittorrent," in *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.
- [13] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proceedings of ACM SIGCOMM'04*, pp. 367–378, August 2004.