A Dependable Cluster-based Topology in P2P Networks

Guiyi Wei, Yun Ling, Ye Gu, Yujia Ge
School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou, P.R. China
Email: {weigy, yling}@zjgsu.edu.cn, zhuge011@126.com, yge@zjgsu.edu.cn

***Abstract*—Unstructured peer-to-peer network is a prevalent model in current P2P networks. In general, unstructured P2P model divides the sharing file into many chunks. A peer must search the positions before downloads a chunk. Since a peer does not know the global topology of the overlay network and the distributions of the wanted chunks, what he can do is flooding chunk queries without a proper order when search and download chunks. Additionally, some peers may behavior selfish: leaves network after completed downloading, downloads but not (or limit) its upload. So, the heavy-tail (or long-tail) phenomenon always exists in the unstructured peer-to-peer networks. Some peers cannot complete their download tasks for the scarcity of certain chunks. By combining topology-control and priority-order, this paper proposes a cluster-based model solve the heavy-tail problem. It analyzes the distribution of file chunks and increases the replication speed of the rarest chunks transmitted among the group. Our simulation experiments show the proposed model provide a dependent network topology for a P2P system, and it helps to transmit the rarest chunk efficiently and improve the overall download completion rate.**

***Index Terms*—unstructured P2P network, heavy-tail, cluster, topology control, complete rate**

I. INTRODUCTION

An unstructured and decentralized P2P system has the following characteristics: any two peers can communicate with each other; any peer can attend or leave the interesting swarm freely and dynamically; peers are completely equal; nodes share some resources each other. The performance of an unstructured and decentralized P2P system relies on the users' willingness to contribute their bandwidth. Since a peer does not know the global topology of the overlay network and the distributions of the wanted chunks, what he can do is flooding chunk queries without a proper order when search and download chunks. Additionally, some peers may behavior selfish: leaves network after completed downloading, downloads but does not upload. Thus, the heavy-tail (or long-tail) phenomenon always exists in the unstructured peer-to-peer networks. Some peers cannot complete their download tasks for the scarcity of certain chunks. Due to the substantial characters of the overlay network and some peers' selfish behaviors, the distributions of file pieces will not be uniform. Some chunks may have lots of copies in the network while others are sparse. Extremely, some chunks may vanish. This will result in a decline in downloaded completion rate. When downloading a file, many users often encounter a downloaded file with 99% and its download speed is 0. The above problem is called the heavy-tail (or long-tail) problem.

Heavy-tail phenomena have been observed in many natural phenomena including both physical and sociological phenomena. In P2P file-sharing system, heavy-tail means that when someone wishes to obtain less popular content, he may often have to wait in the swarm indefinitely to meet peers that have the chunks he needs. It also means that a small number of peers (seeds) own all chunks and most peers own little chunks which can't form a complete file. Once seeds leave the system, other peers can't get all chucks.

To solve the heavy-tail problem, BitTorrent uses a tit-for-tat incentive strategy to motivate users to provide higher upload bandwidth [2]. It is an iterative process of Prison's Dilemma. Winners are users with most contributions. But the process is difficult to execute restrictively in practice [3]. The proposed cluster-based model in this paper is to solve the heavy-tail problem and make file chunks distribute as evenly as possible in the network. The basic principle is to increase copies of rare chunks in global. The model puts the users which download the same file into one cluster. It identifies rare chunks by detecting the situation of chunks owned by other clusters and transmits the chunks to each peer in the cluster. The purpose of this model is to increase the completion rate. The main keys in the model are: 1) How to organize the nodes into clusters, and maintain a proper cluster size? 2) How to determine the distribution of chunks?

In this paper, we first describe the formation of clusters. Then we use a detecting method similar to Random Walk routing algorithm, to learn chunk distributions in other clusters. The contributions in this paper include: 1) we propose a new cluster-based model, which dynamic merges and divides peer clusters to keep proper cluster size. The size of cluster is based on the linking ability of users, and it also estimates the size of the network in a simple way. 2) During the downloading process, the rarest resources are identified by detecting the situations in other clusters. Experiments show the proposed cluster based model can efficiently improve the download completion rate.

The rest of this paper is organized as follows: Section 2 discusses the related work. Section 3 presents our cluster

Manuscript received April 15, 2009; revised October 15, 2009; accepted December 1, 2009.

doi:10.4304/jcm.5.1.57-62

based model and a method which determines how to form a cluster and how to configure the number of the members in a cluster. Section 4 uses a detection method to obtain a distribution of chunks and determine which chunk should be given high priority. Some simulations are shown in Section 5 and we conclude in Section 6.

II. Related Work

To accurately understand P2P systems, many researchers have tried to model the behaviors of the peers and the network. Li et al. [7] propose a Dynamic Layer Management algorithm, DLM, which can maintain an optimal layer size ratio and adaptively elect and adjust peers between super layer and leaf-layer. Jun [8] has focused on the fairness, robustness, and performance characteristics of BT, mainly resulting from the tit-for-tat mechanism. Landa et al. [10] present PledgeRoute, an accounting mechanism for peer contributions that is based on social capital. The system is resistant to three kinds of attacks, and operates as an incentives mechanism by ensuring higher service quality for those peers that contribute more resources to the overlay. Ramachandran et al. [18] propose a problem of Blocked Leecher Problem (BLP). It is peers attempting to download such a file may have to wait indefinitely to obtain certain file chunks that are not distributed in the file's network of peers. To alleviate BLP, they propose BitStore, a larger, secure network of BitTorrent users (not necessarily all sharing the same content) where nodes offer their resources (such as disk space and bandwidth) for public use. BitStore also provides robust incentives for nodes contributing resources: In return for storing and serving chunks, such nodes can negotiate micro-payments using a second-price auction. A number of research studies (e.g., [1], [4], [5], [6], [8], [14], [15], [16], [17]), have focused on the fairness, robustness, and performance characteristics of BT, mainly resulting from the TFT mechanism. In [16] a fluid model is described to overcome the computation problem in [17]. Seeds are a common phenomenon seen in the BT-like system. However, very few studies have considered seeds behavior effects. Chow et al. [12] propose a simple and scalable approach that makes more intelligent use of seed capacity by hurting free-riders, without their explicit identification, while improving the performance of contributing nodes. Terpstra et al. [11] propose a simple probabilistic search system, BubbleStorm, built on random multi-graphs. The approach is a flexible and reliable strategy for performing exhaustive search. In [13] a CDC (Connectivity-based Distributed Node Clustering) scheme is proposed to discover connectivity-based cluster and handle the node dynamics in peer network. Their experiment shows that utilizing message-based connectivity structure can considerably reduce the messaging cost and provide better utilization of resources. Although we also utilize cluster to manage peers, the objective and cluster of formation are different from [13]. Ezovski et al. [21], by using the water-filling technique, determine how each peer should use its capacity to sequentially minimize the file download times in an upload-constrained P2P network. Chan et al. [22] pay attention to investigate the data distribution problem, and present the graph-based dynamically weighted maximum-flow algorithm which is addressing this collaborative file distribution problem and formally define the scheduling problem in a simplified context.

III. Cluster-based Model

In an unstructured P2P download, we can often meet the 99% phenomena since a P2P system is different from a traditional server/client model, in which each node is both a service provider and a user. That is, we can download file chunks from a seed node, and also from other peers. If a seed leaves, the completion of a file download needs the help from peers cooperatively. While some rational users usually are not willing to upload files for others, there exist some nodes with 99% completion rate and 0 kb/s download speed. That is, when a certain file chunk is missing in the network, all the peers cannot complete the download. The objective of the cluster based model is to make the chunk distribution more even and let the rarest chunk not only held by rational users in the network.

A. Formation and division of clusters

In this section, we describe how a cluster is formed and divided. Let us define two concepts: *steady* state and *saturated* state. A *saturated* state refers to a cluster with maximum members among which each node has already had its maximum connections. N is the number of nodes in this state. A *steady* state has a half number of nodes in a saturated state. A steady state also means this cluster can still accept or be accepted by any other node. We use n to represent the number of nodes in a steady state. When a saturated cluster accepts a node, it can be divided into two steady clusters.

(1)Formation of a cluster.

The first peer of a shared file in the network builds the first cluster. Other peers with same interests query to discover existed peers. The formation process is as follows.

- When there is only one seed at first, the seed distributes messages in a limited flooding mechanism and shows that file F with size of m can be downloaded.
- Before downloading a file, a node first searches the file using a flooding mechanism. The nodes using the flooding messaging mechanism then form a cluster by message collisions. At startup, there are only 2 nodes: one seed and one peer in Figure 1(a).
- The cluster does not reach a saturated state yet.
- Assume the steady cluster has n nodes. If other r nodes want to download file F, where $r+2 = n$. Due to the stochastic property of networks, these nodes can form a new cluster, or combine with previous clusters, or divide into several smaller clusters. Let us consider the different situations separately. 1)

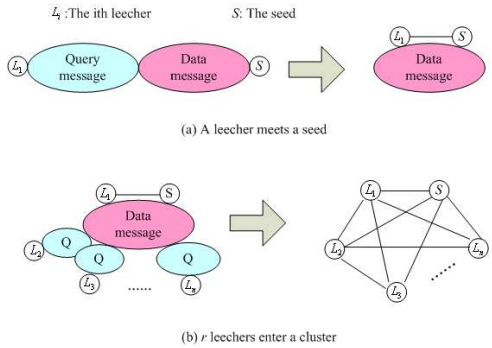

Fig. 1. Formation of a cluster.

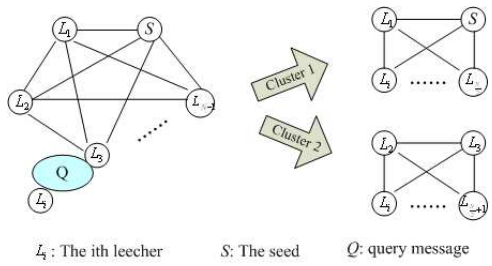

Fig. 2. Division of a cluster.

Combine with previous clusters. It reaches a steady state and then does not send any flooding message to the outside. Nodes in the cluster connect with each other. 2) Form a new cluster. If the r nodes form a new cluster, the cluster needs to continue messaging for accepting new members since $r < n$. It will later reach the state as in the previous step.

The process of cluster formation is shown in Figure 1. The cluster in this paper isn't like the cluster on literature [7] which have super peer whose duty is transferring flooding query message. The dashed line represents the omitted peers in Figure 1. we use the simplest search methods, although many researches have proposed state-of-the-art methods for resource discovery in the P2P networks, such as Terpstra et al. [11] introduces a new communication primitive, called Bubblecast, to complete the query process, and Feng et al. [9] makes a difference between different "statuses" of nodes according to the properties of a discovery network.

(2)Division of a cluster.

When a new node comes, we will let it join a certain cluster first and assign some upload bandwidth to it. When the connections of nodes reach the constraints, the cluster will be divided into two new clusters after accept the new comer. As shown in Figure 2, when a node wants to join a saturated cluster, the cluster will be divided into two steady clusters. If N is odd, the size of two new clusters are both $(N+1)/2$. If N is even, the size will be $\frac{N}{2}$ and $\frac{N}{2}+1$ respectively.

Four types information sets are recorded to preserve the peer relationship in a cluster. (a) *Basic information* consists of *Cluster Id*, *Filename to Download*, *Maximum Connection*, *Number of Neighbors inside a Cluster*, *Joining Time*, and *Number of Clusters*; (b) *Neighbors insider a cluster* consists of *Filename to Download*, *Address of Neighbors Inside a Cluster*, and *Joining Time of inside Neighbors*; (c) *Neighbors outside a cluster* consists of *Filename to Download*, *Cluster ID*, *Address of Neighbors outside a Cluster*, and *Joining Time of Outside Neighbors*; and (d) *Neighbors update information consists of Filename to Download*, *Cluster ID*, and *Address of outside Neighbors*.

While the connection to the new comer is accepted, the information of neighbors in the cluster is sent to this node and information of outside neighbors is exchanged. If a new cluster is found and connected, the information of updated neighbors must be sent to all the nodes of the cluster. If the connection issued by the new comer is rejected by the cluster, the information of the outside neighbors of the cluster should be sent to the new comer to help it connect to other clusters.

B. Estimate network size of file downloads approximately

According to the model described above, it is possible to estimate the downloading network size when the first seed exists. There is only one cluster when a network starts. Due to the nodes are connected with each other, it means all the nodes have connection with the seed and seeds know the connection time of each node. According to *Little's result*, $\lambda T = N$, where N is the average number of nodes, λ is the number of nodes joining the cluster during a time unit, and T is the average time of each node staying in a cluster. And $T = \frac{\sum_{i=1}^{n_c} T_i}{n_c}$, where n_c is the number of nodes in the current cluster. We can derive λ from putting Equation (2) into Equation (1). Let T' denote the time for a seed to upload the file to a node, so the completion time for all the downloads will be no less than T'.

After the duration of T', some nodes may leave after its download completes. We can estimate N accordingly. When the first division happens, the existing number of nodes is recorded. We can also estimate the number of clusters after the division. Assume the number of cluster is h, $h = \frac{N}{N_c}$, where N_c is the number nodes in the cluster when division happens.

IV. Detection method

Peers will download chunks owned by the cluster as a high priority. Chunk information is exchanged from time to time. This is first for ordering downloaded chunks and secondly for detecting chunk information in other clusters conveniently.

A. Chunk tables

For downloading the rarest resource as a high priority, a *chunk table* is designed to store chunk information, which includes Download-File-Name, Chunk-ID, Chunk-Priority, Number-of-Owned-Chunks, and Rarest-Chunk-ID. In the chunk information set, chunk priority is to determine the downloading order. The smaller the priority value is, the earlier the download happens. This is to

let nodes have same chunks, which can increase the download speed. If two chunks have same priority, one is chosen randomly. From the beginning, every chunk has a big priority number. After the chunk is downloaded once, the priority is decreased by 1 and the number of owned chunks is increased by 1. If chunk change information (include Download-File-Name and Completed-Chunk-ID) from other node is received, the corresponding priority of this chunk is also decreased by 1.

For example, in Cluster 1, there are 5 nodes which are A1, A2, A3, A4 and A5 as shown in Table 1. The rarest resource column is not listed. We can learn that the downloading file is divided into 6 chunks. Initial priority for each chunk is 100. The summary is shown in Table 2.

From Table 2, node A1 requests downloading Chunk 3, 4, 5, and 6 according to their priority. A1 downloads Chunk 3 first. Because of Chunk 3's low priority, its download speed is fast. Since not all the download tasks can be complete in Cluster 1, it needs communications to outside for seeking resources. The processes of looking for resources and analyzing chunk distribution in some other clusters are combined. That is, when looking for a missing chunk, the statistics of distributions of chunks in other clusters are also performed at the same time.

B. The statistics of chunk distribution

Let seeds or nodes having all the chunks inside the cluster detect chunk distribution in other clusters. The nodes can send detection messages to the addresses of *Neighbors outside a cluster*. For not transmitting its determination of rare chunks, the detected nodes in other clusters only return their first 4 items of chunk table. After receiving chunk tables, we can calculate the rareness value of a chunk according to the principles as follows: 1) if a certain chunk exists in every cluster, its rareness value is low, 2) if the number of existing copies of two chunks are the same, the dispersion is considered. The rareness value of a chunk with scattered copies is smaller than centralized ones. We use Equation (1) to calculate the rareness value of a chunk. The chunk rareness values of all chunks are stored as auxiliary information and dynamically changed.

$$F(m) = \sum_{i}^{h-1} (au_i + v_i) \tag{1}$$

Here, a is a large constant, h is the number of clusters, m is chunk ID, u_i represents different clusters which means the dispersion, with the value of 0 or 1, and v_i represents the number of chunks in a cluster.

When one chunk's u_i is 0, v_i is also 0 accordingly. When one chunk's u_i is 1, v_i is the value derived from the maximum priority deducted by current priority. Equation (1) tells us that if the rareness value is low, $F(m)$ is high. Therefore, what we need is the chunk with the smallest $F(m)$.

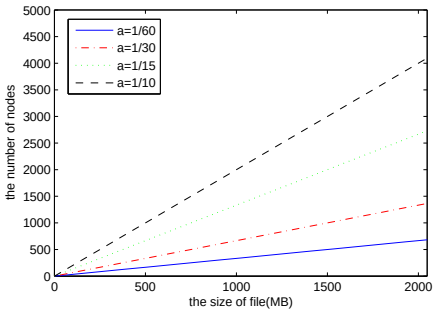

Fig. 3. Size of the P2P network according to the parameter α.

Assume there are 2 clusters: cluster 1 and cluster 2. Cluster 2 sends detection message to A1 in cluster 1. Then A1 return 4 items of its chunk table to nodes in Cluster 2. If $a = 10000$, we have $F(1) = 10004$, $F(2) = 10004$, $F(3) = 10003$, $F(4) = 0$, $F(5) = 0$, and $F(6) = 0$.

After getting the rareness value outside the cluster, the rareness values are distributed to all the nodes in the cluster. In the above example, chunk 4, 5 and 6 are rare in cluster 1. The rarest chunk item in the cluster information set is filled by 1. Nodes in this cluster should download these chunks first to reach the goal for distributing rare resources.

V. Simulation Experiment

To prove the efficiency of our cluster based model, assume there are 10000 users with same hardware configurations to download a file of 1GB. One user joins the network in every 1 minute. The maximum upload bandwidth is 50KB/s and maximum connection is 100. According to Equation (1), there are 334 nodes in the network. They should be divided into 3 groups. The size of file has direct ratio with the number of existing nodes in the network shown in Figure 3. Where, α represents the number of nodes joining the cluster in a time unit. The bigger α is, the larger the network size is.

In the second experiment, we compare the download speed of peers and free-riders. Free-riders are 20% of total peers. By analyzing the fluctuation of download speed of a certain node, the cluster based model has higher download speed than the tit-for-tat model in Figure 4. Since files which have been held in the cluster are downloaded first in the cluster based model, the download speed is very fast at first. In the middle of a download, the speed keeps 0.3 chunks/s. When at the end of a download, since the copies of rare chunks increase, the speed does not decrease as radically as in tit-for-tat strategy. As to the tit-for-tat strategy, the download speed is low both at the beginning and end of a download. The average speed is 0.23 chunks/s.

Experiments for downloading files with different size are also conducted. Comparison result of their download speed of tit-for-tat strategy and cluster based model is shown in Figure 5. In the situations of 100MB, 500MB and 1000MB, download time in cluster based model is all shorter than tit-for-tat.

TABLE I
AN EXAMPLE OF CHUNK TABLES. (CID="CHUNK ID", PR="CHUNK PRIORITY", AND OW="NUMBER OF OWNED CHUNKS"

A1			A2			A3			A4			A5		
Cid	Pr	Ow	Cid	Pr	Ow	Cid	Pr	Ow	Cid	Pr	Ow	Cid	Pr	Ow
1	96	1	1	96	1	1	96	0	1	96	1	1	96	1
2	96	1	2	96	1	2	96	1	2	96	1	2	96	0
3	97	0	3	97	1	3	97	1	3	97	1	3	97	0
4	100	0	4	100	0	4	100	0	4	100	0	4	100	0
5	100	0	5	100	0	5	100	0	5	100	0	5	100	0
6	100	0	6	100	0	6	100	0	6	100	0	6	100	0

TABLE II
DATA SUMMARY

Node	Owned Chunks	Needed Chunks	Downloaded Chunks
A1	1, 2	3, 4, 5, 6	3
A2	1, 2, 3	4, 5, 6	Need outside chunks
A3	2, 3	1, 4, 5, 6	1
A4	1, 2, 3	4, 5, 6	Need outside chunks
A5	1	2, 3, 4, 5, 6	2

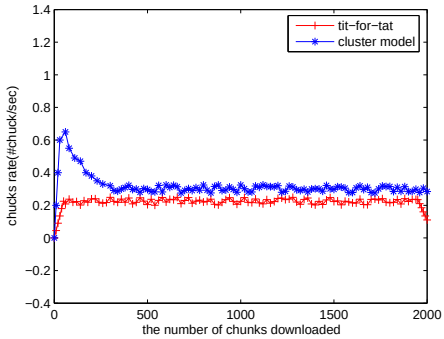

Fig. 4. Comparison with tit-for-tat on chunks distribution.

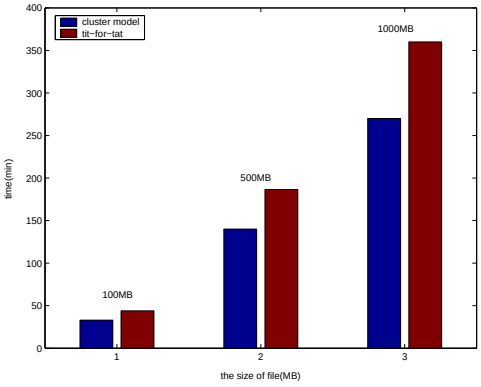

Fig. 5. Comparison with tit-for-tat on complete time.

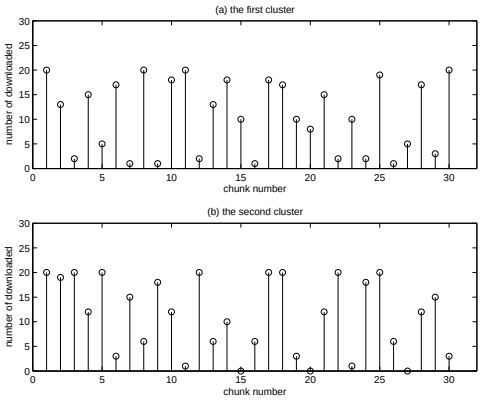

Fig. 6. Number of downloaded chunks before the detection.

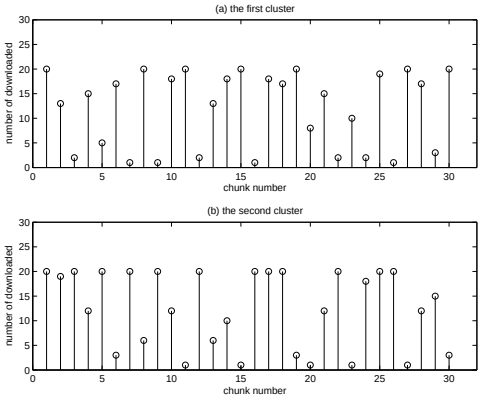

Fig. 7. The number of downloaded chunks after the detection.

The third experiment compares the distribution of rare chunks before and after detection. Suppose there are two clusters, each with 20 nodes. They are downloading the same file with 30 chunks. The chunk tables before and after detection are analyzed. The downloaded chunks are shown in Figure 6. Suppose there is no new node joining the clusters during this time.

From Figure 6, since there is no chunk with 0 number of downloaded (the downloaded number in seed node is regarded as 1 by default), there exist seed nodes in cluster 1. Cluster 2 does not have a seed node since there is some chunk with 0 number of downloaded, and it cannot complete the file download within cluster 2. After doing detection in these two clusters and transmitting the rare chunk information, the situation changes as shown in Figure 7.

From Figure 7, cluster 1 finds out that cluster 2 is lack of chunk 15, 20, 27. Cluster 1 downloads these chunks at high priority to increase the replications of these chunks. In Figure 7(a), the downloaded of chunk 15, 20 and 27 reaches 20. When cluster 2 learns the rarest chunks in

cluster 1 is chunk 7, 9, 16, 26, cluster 2 downloads these chunks. Figure 7 (b) shows the downloaded number of chunk 7, 9, 16 and 26 reaches 20.

Since the strategy in our model is to download the chunks held in the cluster, we can use it to find free-riders. For example, if one chunk has been downloaded by many nodes but its download speed is still very low, it is high possibility of the existence of free-riders.

In our proposed model, if there is no network congestion, the download speed of a peer i is

$$d_i = \frac{\sum_{j=1} N_s u_S^j}{N_l} + \frac{\sum_{k=1} N_l u_d^k}{N_l - 1}. \qquad (2)$$

Where $\sum_{j=1} N_s u_S^j$ and $\sum_{k=1} N_l u_d^k$ represent the upload speed of seeders and peers, N_l is the number of peers. Assume the maximum download and upload speed is D_i and U_i, respectively. The download and upload speed of a user is asymmetric, since people usually set the upload speed is smaller than download speed. Referred to the results of literatures [4, 5, 6], it is reasonable $u_i = U_i \leq D_i$ since users can limit their upload speed in most P2P systems.

Suppose U_s and N_l do not change in a small duration of time. When downloading a chunk with similar priority, d_i becomes smaller. We think the decrease of u_d causes d_i to change be smaller. The underlying possible reason is that there are many free-rider with $u_d = 0$ connected with the node. A comprehensive research on free-riders using chunk tables are our future work.

VI. Conclusion

This paper proposes a cluster based model to solve the heavy-tail problem. By combining topology-control and priority-order, the proposed model increase the replications of rare chunks in the network and make file chunks distribute evenly. Chunk tables are designed to analyze the chunk distribution outside the cluster and let nodes in a cluster download the chunks lacked in other cluster. We also estimate a network size roughly and estimate the cluster numbers quantitatively. The simulation experiments show the cluster based model can make file chunks distribute more evenly and increase the download completion rate. The results of this paper are useful for the unstructured and decentralized P2P network systems to increase the completion rate, especially helpful to some un-popular file sharing swarms.

Acknowledgment

This work was supported in part by China NSF under Grants 60673179 and Science Foundation of Zhejiang Province of China under Grant number Z106727.

References

[1] D. Hughes, G. Coulson, and J. Walkerdine, "Free Riding on Gnutella Revisited: The Bell Tolls?" IEEE Distributed Systems Online, 6(6), June 2005.

[2] B. Cohen, "Incentives Build Robustness in BitTorrent", P2PEcon, June 2003.

[3] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. "Exploiting Bittorrent for Fun (But Not Profit)", IPTPS, February 2006.

[4] S. Saroiu, P.K. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-To-Peer File Sharing Systems", Multimedia Computing and Networking 2002 (MMCN 02), San Jose, USA, January 2002.

[5] B. Fan, D.-M. Chiu, and J.C. Lui, "The Delicate Tradeoffs In Bittorrent-Like File Sharing Protocol Design", Prof. of the 2006 IEEE International Conference on Network Protocols(ICNP), Washington, DC, USA ,2006, pp. 239-248 .

[6] A.Chow, L.Golubchik, V.Misra, "BitTorrent: An Extensible Heterogeneous Model", the 28th Annual Conference of the IEEE Communications Society (INFOCOM'09), Rio de Janeiro, Brazil, April 2009.

[7] X. Li, Z.Y. Zhuang, Y.H. Liu, "Dynamic Layer Management in Superpeer Architectures", IEEE Transactions on Parallel and Distributed Systems, 16(11), November 2005, pp. 1078-1091.

[8] S. Jun, M. Ahamad, "Incentives In Bittorrent Induce Free Riding", the proceedings of P2PEcon'05, 2005.

[9] G. F. Feng, J. C. Zhang, Y. Q. Jiang, et al. "Optimization of Overlay Topology for Unstructured Peer-to-Peer", Journal of Software, 18(11), November 2007, pp.2819?2829.

[10] R. Landa, D. Griffin, R. G. Clegg, et al. "A Sybilproof Indirect Reciprocity Mechanism for Peer-to-Peer Networks", the 28th Annual Conference of the IEEE Communications Society (INFOCOM'09), Rio de Janeiro, Brazil, April 2009.

[11] W.W. Terpstra, M.K. Raxnanathan, A. Aware, et al. "BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search", Proc of SIGCOMM'07, Kyoto, Japan, August 2007, pp.27-31.

[12] A. L. Chow, L. Golubchik, and V. Misra, "Improving bittorrent: A simple approach", Proc. of IPTPS'08, 2008.

[13] L. Ramaswamy, B. Gedik, L.Liu. "A Distributed Approach to Node Clustering in Decentralized Peer-to-Peer Networks", IEEE Transactions on Parallel and Distributed Systems, 16(9), Sept. 2005: 814 - 829.

[14] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, "Do incentives build robustness in bittorrent?" , Proc. of NSDI'07, 2007.

[15] M. Sirivianos, J.H. Park, R. Chen, X. Yang. "Freeriding in bittorrent networks with the large view exploit", Proc. of IPTPS'07, 2007.

[16] D. Qiu and R. Srikant. "Modeling and performance analysis of bittorrentlike peer-to-peer networks", Proc. ACM SIGCOMM'04, 2004.

[17] X. Yang and G. de Veciana. "Service capacity of peer to peer networks", Proceedings of IEEE INFOCOM'04, 2004.

[18] A.Ramachandran, D. Sarma, N. Feamster, "BitStore: An Incentive?Compatible Solution for Blocked Downloads in BitTorrent" , Proc. Joint Workshop on The Economics of Networked Systems and Incentive-Based Computing (NetEcon+IBC), San Diego, CA, June 2007.

[19] N. Sarshar, P. N. Boykin, V. P. Roychowdhury, "Percolation search in power law networks: making unstructured peer-to-peer networks scalable", Proc. of IEEE P2P 2004, Pages 2-9, 2004.

[20] R. A. Ferreira, M. K. Ramanathan, A. Awan, et al, "Search with probabilistic guarantees in unstructured peer-to-peer networks", Proc of P2P'05, Washington DC: IEEE Computer Society, 2005: 165-172.

[21] G. M. Ezovski, A. Tang, L. L.H. Andrew. "Minimizing Average Finish Time in P2P Networks", the 28th Annual Conference of the IEEE Communications Society (INFOCOM'09), Rio de Janeiro, Brazil, April 2009.

[22] J. Chan, V. Li, K. Lui. "Performance comparision of scheduling algorithms for peer-to-peer collaborative file distribution", IEEE Journal on Selected Areas in Communications, 25(1):146-154, January 2007.

Guiyi Wei received the B.Sc and M.Sc degrees in Information Management and Information System from Zhejiang Gongshang University, China. in 1996 and 2000 respectively, and Ph.D. degree from Zhejiang University, China, in 2006 from Computer Science. Now he is an Associate Professor in the Department of Computer Science and Information Engineering of Zhejiang Gongshang University, China. His research interests include grid/cloud computing, peer-to-peer networks, and wireless sensor networks. He is a member of the IEEE Computer Society.