MODELING, SIMULATIONS, AND EXPERIMENTS TO BALANCE PERFORMANCE AND FAIRNESS IN P2P FILE-SHARING SYSTEMS

by

YUNZHAO LI

B.E., Jinan University, China, 1995

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the

requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2013

Abstract

In this dissertation, we investigate research gaps still existing in P2P file-sharing systems: the necessity of fairness maintenance during the content information publishing/retrieving process, and the stranger policies on P2P fairness.

First, through a wide range of measurements in the KAD network, we present the impact of a poorly designed incentive fairness policy on the performance of looking up content information. The KAD network, designed to help peers publish and retrieve sharing information, adopts a distributed hash table (DHT) technology and combines itself into the aMule/eMule P2P file-sharing network. We develop a distributed measurement framework that employs multiple test nodes running on the PlanetLab testbed. During the measurements, the routing tables of around 20,000 peers are crawled and analyzed. More than 3,000,000 pieces of source location information from the publishing tables of multiple peers are retrieved and contacted. Based on these measurements, we show that the routing table is well maintained, while the maintenance policy for the source-location-information publishing table is not well designed. Both the current maintenance schedule for the publishing table and the poor incentive policy on publishing peers eventually result in the low availability of the publishing table, which accordingly cause low lookup performance of the KAD network. Moreover, we propose three possible solutions to address these issues: the self-maintenance scheme with short period renewal interval, the chunk-based publishing/retrieving scheme, and the fairness scheme.

Second, using both numerical analyses and agent-based simulations, we evaluate the impact of different stranger policies on system performance and fairness. We explore that the extremely restricting stranger policy brings the best fairness at a cost of performance degradation. The varying tendency of performance and

fairness under different stranger policies are not consistent. A trade-off exists between controlling free-riding and maintaining system performance. Thus, P2P designers are required to tackle strangers carefully according to their individual design goals. We also show that BitTorrent prefers to maintain fairness with an extremely restricting stranger policy, while aMule/eMule's fully rewarding stranger policy promotes free-riders' benefit.

MODELING, SIMULATIONS, AND EXPERIMENTS TO BALANCE PERFORMANCE AND FAIRNESS IN P2P FILE-SHARING SYSTEMS

by

YUNZHAO LI

B.E., Jinan University, China, 1995

A DISSERTATION

submitted in partial fulfillment of the

requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2013

Approved by:

Co-Major Professor

Don Gruenbacher

Approved by:

Co-Major Professor

Caterina Scoglio

Abstract

In this dissertation, we investigate research gaps still existing in P2P file-sharing systems: the necessity of fairness maintenance during the content information publishing/retrieving process, and the stranger policies on P2P fairness.

First, through a wide range of measurements in the KAD network, we present the impact of a poorly designed incentive fairness policy on the performance of looking up content information. The KAD network, designed to help peers publish and retrieve sharing information, adopts a distributed hash table (DHT) technology and combines itself into the aMule/eMule P2P file-sharing network. We develop a distributed measurement framework that employs multiple test nodes running on the PlanetLab testbed. During the measurements, the routing tables of around 20,000 peers are crawled and analyzed. More than 3,000,000 pieces of source location information from the publishing tables of multiple peers are retrieved and contacted. Based on these measurements, we show that the routing table is well maintained, while the maintenance policy for the source-location-information publishing table is not well designed. Both the current maintenance schedule for the publishing table and the poor incentive policy on publishing peers eventually result in the low availability of the publishing table, which accordingly cause low lookup performance of the KAD network. Moreover, we propose three possible solutions to address these issues: the self-maintenance scheme with short period renewal interval, the chunk-based publishing/retrieving scheme, and the fairness scheme.

Second, using both numerical analyses and agent-based simulations, we evaluate the impact of different stranger policies on system performance and fairness. We explore that the extremely restricting stranger policy brings the best fairness at a cost of performance degradation. The varying tendency of performance and

fairness under different stranger policies are not consistent. A trade-off exists between controlling free-riding and maintaining system performance. Thus, P2P designers are required to tackle strangers carefully according to their individual design goals. We also show that BitTorrent prefers to maintain fairness with an extremely restricting stranger policy, while aMule/eMule's fully rewarding stranger policy promotes free-riders' benefit.

Acknowledgments

I forever appreciate for my advisors Dr. Don Gruenbacher and Dr. Caterina Scoglio. I am very proud for receiving their continuous and inspiring guidance for the past six years. They not only teach me how to do research step-by-step, but also encourage me how to bravely face the difficulties in my life.

I would also like to give my thanks to my committee members Dr. David Soldan, Dr. Xinming Ou, and Dr. Todd Easton. Their direct assistance and insightful comments open me to new research ideas.

Moreover, my thanks also go out to my colleagues Phillip, Hong, Sakshi, Ling, Su and other Sunflower networking group members who, though not providing direct assistance for this work, create a warm, friendly working environment and share research experience with me.

Finally, I would like to thank my parents Xueli Liand Shumei Yi, my wife Lei Fan and my daughter Carol for their patience, support, encouragement and unconditional love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we briefly introduce the P2P file-sharing systems and their unique features. After that, we present the fairness challenges in P2P file-sharing systems and review the related work. Then, we show the motivation of this thesis and conclude with our contribution.

1.1 P2P File-Sharing Systems

Peer-to-peer (P2P) architecture is a distributed application architecture that assigns services and workloads among peers. P2P file-sharing systems, as the most popular application using P2P technology, generate a large amount of the traffic on the current Internet [1]. In a typical P2P file-sharing system, users are called peers, and they run P2P client applications to join the system through the Internet. Each user can connect to other peers for exchanging content like movies, music, and games.

P2P file-sharing systems employ several unique features, which can help peers exchange content effectively:

- each peer has dual roles as both server and client,
- the logical structure of P2P file-sharing systems is decentralized,
- and the management style of P2P file-sharing systems is self-organized.

Compared with traditional client-server file-sharing systems, the dual roles of a peer bring higher throughput and larger scalability to P2P file-sharing systems. In traditional client-server file-sharing systems like FTP, all clients are just consumers, and they must download content from a certain number of pre-designated servers. Therefore, client performance will degrade as the number of users increases, since the fixed service throughput must be allocated among all users. However, in P2P file-sharing systems, each peer acts as both client and server. As a client, it can download content from other peers; as a server, it can also upload content to others at the same time. Thus, the increase in the number of users not only brings larger workloads, but also leads to higher throughput in P2P file-sharing systems. The P2P decentralized logical structure avoids the deadly effect of central service nodes failures and consequently leads to increased robustness. On the other hand, due to the lack of the central nodes, self-organization management style is naturally employed into P2P file-sharing systems. That is, instead of centrally collecting global information of the whole system and then using the information to manage each peer, any peer in a typical P2P file-sharing system will individually decide how to exchange content with other peers according to their own experience. As a result, self-organization also brings additional challenge into P2P file-sharing systems like the requirement of maintaining fairness.

1.2 Fairness Challenges in P2P File-Sharing Systems

Maintaining fairness is necessary in P2P file-sharing systems, because 1) system performance fully relies on the cooperation of each peer, and consequently peers' selfish behavior will reduce system performance, and 2) special peers called "free-riders" always exist in the system. Compared with general peers existing in a P2P file-sharing system, who obey the P2P principle to act as both service providers and service consumers, free-riders are only service consumers, who spend

the downloading resource without any contribution to other peers. Previous research already showed that a large number of free-riders can be discovered in P2P file-sharing systems [2]. The popularity of free-riding comes from the decentralized infrastructure and self-organization of P2P networks. Without central management nodes, each free-rider can freely choose to partner any number of peers and can quickly change partners for avoiding punishment. Obviously, since the systems performance of P2P file-sharing networks relies on the cooperation of each peer, free-riding behavior contributes nothing to system performance; moreover, it is unfair to contributors and eventually degrades system performance. Therefore, each P2P file-sharing system must disincentivize this free-riding behavior and provide fairness to general peers.

Fairness management in P2P file-sharing systems is also restricted by its decentralized infrastructure and self-organization. Thus, a typical P2P file-sharing system usually adopts incentive policies instead of obligatory strategies. The generous behavior of peers is rewarded by promoting more downloading bandwidth, and the free-riding behavior is punished with the reduction of the downloading rate. As monetary payment-based incentive policies face many implementation issues in a typical P2P environment [3], reciprocity-based incentive policies become the fundamental component of fairness strategies [4, 5]. Under this reciprocity-based incentive mechanism, shareable content or uploading bandwidth is recognized as goods for trading in the system. A peer can then choose their partners for content exchange according to their individual current or previous behavior. Generous partners can obtain higher priority and larger bandwidth than selfish partners.

1.3 Prior Work on P2P Fairness

The free-riding issue of P2P file-sharing systems was first found in [2], where the authors showed that around 70% users of Gnutella P2P file-sharing network [6]

did not share any files. The following study in [7] found 85% of peers in Gnutella were free-riders. Besides Gnutella, free-riders also exist in other popular P2P file-sharing systems like BitTorrent and aMule/eMule. The impact of free-riding on system performance was evaluated in [8, 9, 10].

To restrain the free-riding behavior, the research community suggested and evaluated incentive fairness policies. A number of studies focused on designing the reciprocity-based incentive fairness policy in P2P file-sharing networks. On one side, researchers tried to maintain system fairness based on global knowledge of each peer's behavior information. For example, researchers in [11] suggested that each peer constructed a trust graph covering itself and its target peers, and then the reputation levels of these target peers were computed through different trust paths. Alternatively in [12], the local reputation scores of each peer's partners would first be computed according to their previous behaviors. After that, their global reputation values were determined using a centralized computing algorithm, which calculated the eigenvector of a network trust matrix. In [13], the authors acted to improve system fairness based on an existing social network. In [14], a general analytical framework was proposed to help researchers evaluate different incentive strategies of P2P networks.

On the other side, historical information cannot easily be spread and synchronized within a distributed and self-organized P2P system. In reality, P2P file-sharing systems tend to employ simpler local information based fairness policies. For example, aMule/eMule[15, 16] employs a local credit system to maintain fairness, where a peer, after uploading content to its partners, will be assigned some credit by these partners, and these credit can only be used in pairs between this peer and the direct partners in the future. BitTorrent [17] maintains its fairness by using a simple Tit-For-Tat (TFT) incentive policy, where a peer mainly uploads to others from whom it can download at the same time. However, these policies were shown not to be robust for maintaining fairness. Free-riders could even obtain higher download rates than TFT compliant clients [18, 19]. In [20],

a popular free-riding behavior was also found by crawling aMule/eMule for over 50 days. In [21], the results of real world experiments showed that free-riders in aMule/eMule can obtain similar downloading rate as general peers.

1.4 Motivation of Research

Due to the importance of fairness for P2P file-sharing, a large number of related studies have been conducted. However, the research community needs to investigate several research gaps still existing in two primary areas.

a) The necessity of maintaining fairness during content information publishing and retrieving processes

Exchanging content in P2P file-sharing systems includes two fundamental steps:

- the process of publishing/retrieving content information for P2P file-sharing networks,
- and the process of exchanging real content with other peers.

While maintaining fairness for content exchange has been widely studied, the necessity of maintaining fairness for the content information publishing/retrieving process has been overlooked. The research community should answer the following questions:

- Do peers need to cooperate during the process of publishing/retrieving content information? That is, is maintaining fairness necessary in this process?
- What is the performance impact on the publishing/retrieving process which lacks an effective incentive policy?

With the answers to these questions, P2P designers will develop a deep insight into the fairness issues of the content information publishing/retrieving process. Thus, they may conduct more effective fairness policies.

b) The stranger policies on P2P fairness

When designing a P2P fairness policy, how to effectively deal with strangers has been overlooked. For any peer, strangers are defined to the peers whose previous behavior is unknown by this peer. So, strangers for a particular peer can be categorized into 4 groups:

- legitimate peers who just arrive into the system;
- legitimate peers without any content exchange with this peer;
- free-riders whose free-riding behavior are not known by the peer;
- free-riders who pretend to be strangers by frequently changing their identities.

Since strangers can be either legitimate peers or free-riders, when an uploader allocates upload bandwidth, promoting more bandwidth to its strangers may also provide more benefit to free-riders, while limiting more bandwidth to strangers may limit benefit to legitimate peers. Therefore, when designing a stranger policy, restricting strangers is not necessarily better/worse than rewarding strangers for system performance and fairness. As few quantitative evaluations for different stranger policies currently exist in literature, the research community should answer the following questions:

- How does the treatment policy for strangers affect system performance?
- How does the treatment policy for strangers affect system fairness?
- Does the improved fairness brought by a particular stranger policy also result in the improvement of system performance?

By answering these questions, the research community may find the appropriate trade-off between performance and fairness requirements when designing stranger policies.

1.5 Thesis Contribution and Organization

We study the existing challenges in P2P file-sharing systems discussed above, and we have the following contributions.

In Chapter 3, through a wide range of measurement in the KAD network, we present the impact of a poorly designed incentive fairness policy on the content information lookup performance. The KAD network, which is designed to help peers publish and retrieve sharing content information, adopts a distributed hash table (DHT) technology and combines itself into the aMule/eMule P2P file-sharing network. During the measurements, the routing tables of around 20,000 peers are crawled and analyzed. More than 3,000,000 pieces of source location information from the publishing tables of multiple peers are retrieved and contacted. Based on these measurements, we have the following contributions.

1. We discover that the KAD network has a low lookup performance.

2. We develop a distributed measurement framework which employs multiple test nodes running on the PlanetLab testbed [22]. The entire identity (ID) space of the KAD network is uniformly separated into multiple parts, each of which are measured by an individual PlanetLab test node.

3. We show that the maintenance policy of routing table is well designed. The availability of the routing table is high, and the similarity of routing tables is low. More than 80% of the entries in this table are connectable. The entries of routing tables among peers, who are logically close with each other, are different.

4. We discover that the maintenance policy for the source-location-information publishing table is not well designed. The availability of the publishing table is low. On average, more than 75% entries in this table are stale and cannot be connected.

5. We also reveal that more than 75% peers leave the system within one hour

after publishing their downloaded content into the KAD network.

6. By exploring the implementation of the KAD network, we deduct that both the current maintenance schedule for the publishing tables and the poor incentive policy on publishing peers eventually result in the low availability of the publishing table and accordingly cause poor lookup performance of the KAD network.

7. To deal with these issues, we propose three possible solutions: the self-maintenance scheme with short period renewal interval, the chunk-based publishing/retrieving scheme, and the fairness scheme. Both the strengths and weaknesses of these solutions are also discussed.

In Chapter 4, we also evaluate the impact of different stranger policies on system performance and fairness using both numerical analyses and agent-based simulations. Followings are the key observations.

1. The extremely restricting stranger policy brings the best fairness at the cost of the degradation of performance. The extremely rewarding stranger policy cannot provide the highest performance to the system. Appropriately choosing the intermediate policy can bring the highest performance to the system. The system performance can be improved when the potential contribution of general newcomers is quickly promoted, while free-riders could also obtain benefit.

2. The varying tendency of performance and fairness under different stranger policies are not consistent, and the highest performance and the best fairness of the system cannot be reached simultaneously. Specifically, when the system reaches the best fairness, where a very small fraction of uploading bandwidth is allocated to strangers, free-riders will be significantly restricted and cannot easily stay on the system, while the system performance will be negatively affected due to the delay of general newcomers' potential contribution. When the system reaches the highest performance where at least

some part of the uploading bandwidth is allocated to strangers, free-riders also receive this benefit and survive in the system. Moreover, due to free-riders surviving at this point, good peers may tend to free-ride, since they are rational and receive less benefit than free-riders, or they may directly leave the system due to its unfair treatment. Eventually, these consequences will negatively affect the system performance. Taking this design dilemma into account, P2P designers are required to tackle strangers carefully according to their individual design goals.

3. We conduct cases studies for the current most popular realistic P2P file-sharing systems: BitTorrent and aMule/eMule. Based on the agent-based simulation results, we show that BitTorrent prefers to maintain fairness by sacrificing its performance, while aMule/eMule's fully rewarding stranger policy promotes free-riders' benefit. We verify our discovery by both analyzing the TFT incentive policy of BitTorrent and running experiments in the real aMule/eMule network. Additionally, we simulate the credit incentive policy in aMule/eMule to verify the inconsistent tendency of performance and fairness. Finally, we suggest possible alternative improvements for both of BitTorrent and aMule/eMule.

We organize this dissertation as follows. In Chapter 2, we review the related background of P2P file-sharing systems. The review includes the structured and unstructured overlay infrastructures of P2P file-sharing systems; the most popular P2P file-sharing systems BitTorrent and aMule/eMule; the fairness issue of P2P file-sharing systems; and the incentive fairness policies. In Chapter 3 and 4, we provide our foundational contributions of this research. In Chapter 5, we summarize the entire dissertation and point out the possible future work for P2P fairness design.

Chapter 2

Background

In this chapter, we review the related background for this dissertation. We introduce P2P technologies and P2P unstructured and structured overlay infrastructure. We present the principle of P2P file-sharing systems and the most popular P2P file-sharing systems: BitTorrant and aMule/eMule. After that, we discuss the fairness issue in P2P file-sharing systems and the current corresponding countermeasures.

2.1 P2P Technologies

As a special networking technology, P2P helps a single node build connections with other nodes through the Internet. These nodes are called peers, and they combine together to form a logical network called P2P network. Unlike physical networks, P2P networks have their own logical structured or unstructured topologies.

2.1.1 Main Characteristics of P2P Technologies

P2P networks are designed to provide better performance than traditional client-server networks. The greatest difference between client-server networks and P2P networks is the roles of each node. In a typical client-server network as shown in Figure 2.1.1, each node can be either a client or a server, but not both. Clients can only be service consumers, and servers can only be service providers. Usu-

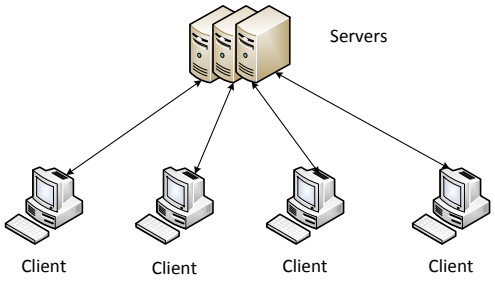


Figure 2.1.1: Client-Server infrastructure

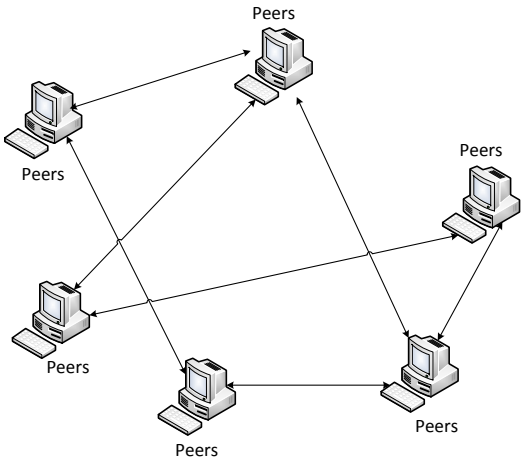


Figure 2.1.2: P2P Infrastructure

ally, servers are pre-designated and have to stay in the system for a long-term period; clients have to obtain service from servers, and there are no communications among clients. In a typical P2P network as shown in Figures 2.1.2, the role of each node is equal: it is both a client and a server. It can obtain service from other peers, and it can also provide service to others at the same time. Thus, communications are distributed among all peers. As a result, the increase in the number of customers in a client-server network usually leads to the degradation of the system performance; while the increase in the number of customers in a P2P network not only brings more workload, it also provides higher service capacity and larger scalability. Furthermore, since each peer can be a service provider, the failures of peers in P2P networks have less impact on the system than the failures of servers in client-server networks. Therefore, the robustness of P2P networks is also stronger than client-server networks.

2.1.2 P2P Overlay Structure

The logical links among peers in a P2P network create an application level based overlay network. Peers use different methods to create their logical links – to discover other peers. Depending on these different methodologies, people can divide this overlay network into two main classes: structured P2P networks and unstructured P2P networks.

2.1.2.1 Structured P2P Networks

Structured P2P networks are normally built by distributed hash table (DHT) algorithms. The hash table algorithm, used to speed up the lookup process, creates the relationship between the index key and the real value. In structured P2P networks, the whole hash table is separated and distributed to all peers. Each peer is responsible for maintaining its own part of the entire hash table. By using maintained information, each peer can also iteratively contact any other peer in the P2P networks. The advantage of structured P2P networks is that it

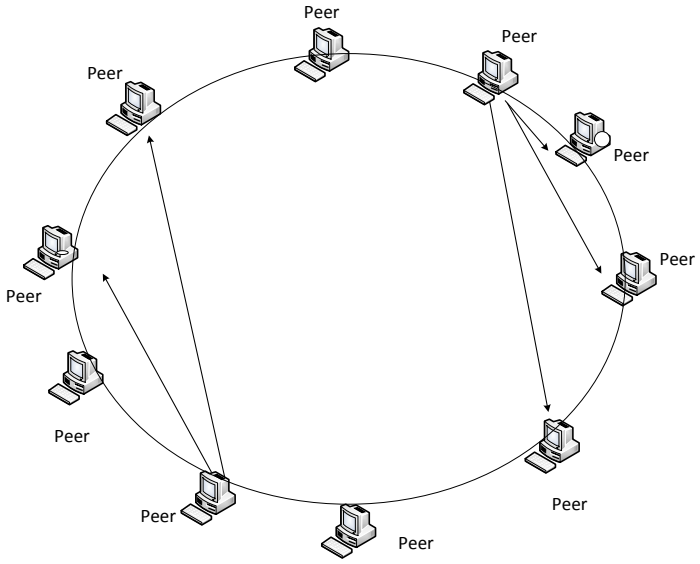


Figure 2.1.3: DHT method - Chord

almost guarantees that online peers can be located within a short-period of time, usually within O($logN$), where N is the population of online peers in the system. Based on how to maintain the neighbors, structured P2P networks have their own logical topologies. For example, the chord system shown in Figure 2.1.3 creates a circle logical overlay topology by letting each peer maintain the information of its successors and predecessors [23]. In the Kademlia network, each peer considers the whole logical system as an unbalanced binary tree [24].

2.1.2.2 Unstructured P2P Networks

In unstructured P2P networks, peers build connections according to an ad-hoc style. To discover other peers in unstructured P2P networks, two different kinds of lookup methods are usually adopted: the index server method shown in Figure 2.1.4 and the query flooding method shown in Figure 2.1.5.

- With the index server method, some central nodes, called index servers, maintain the information of peers. Online peers register themselves on these central nodes and then update this information after some time. Consequently, this information can also be retrieved by any peer from these index

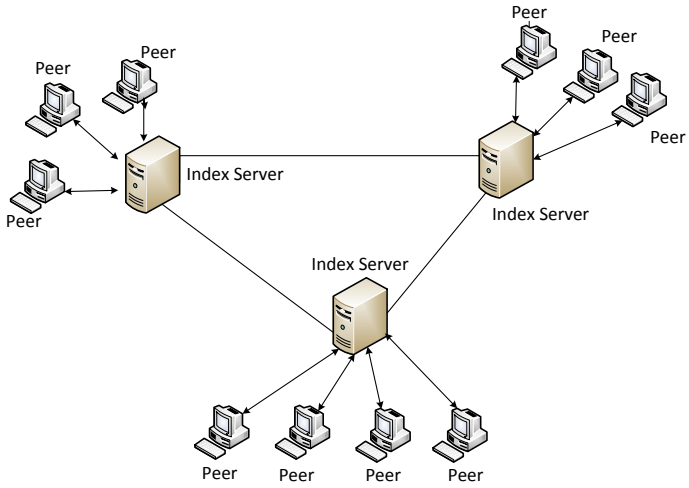


Figure 2.1.4: Index server method

servers. However, this server-based method suffers from the failures of these central nodes, so the scalability and robustness of P2P networks cannot be well extended.

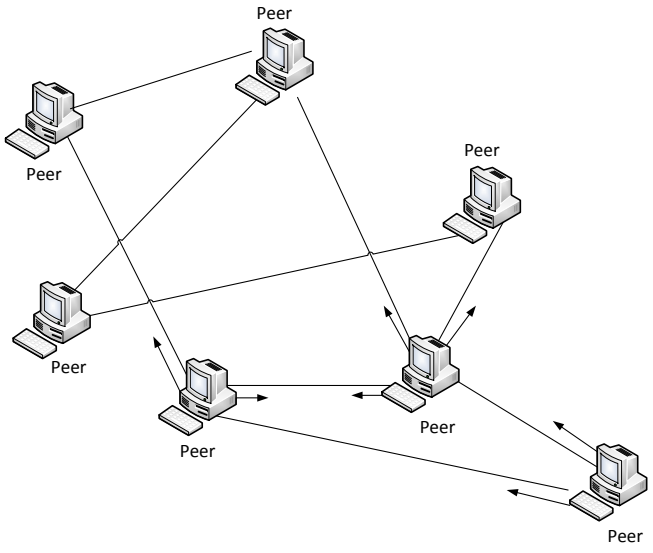


Figure 2.1.5: Query flooding method

- Another lookup method is based on flooding queries among the system. When a peer wants to locate another peer, it will send requests to its known

peers. If these known peers do not have the information of the requested peer, the known peers will iteratively send the requests to their known peers. Obviously, this method could be more robust than the index server method. However, it also introduces a large amount of traffic with the flooding queries, and therefore P2P file-sharing systems usually limit the flooding scope [6]. However, with this limited flooding method, it cannot be guaranteed that the searched peer may be located.

2.2 P2P File-Sharing Systems

P2P file-sharing is the most popular application of P2P technology, which is used for content exchange among peers and produces the largest traffic currently on the Internet [1]. In the following subsection, we introduce general principles of P2P file-sharing systems and the two most popular P2P file-sharing systems: aMule/eMule and BitTorrent.

2.2.1 General Principles of P2P File-Sharing Systems

In P2P file-sharing systems, peers are grouped together for file exchange. Each peer in a P2P file-sharing system normally undertakes two roles: 2) as a service consumer, the peer downloads content from other peers; 1) as a service provider, it also uploads content to other peers. Therefore, P2P file-sharing systems have the following feature: with the increase in the number of joined peers, who are both service consumers and service providers, not only the system downloading workload but also the system uploading capacity increases. When compared with the traditional client-server file-sharing infrastructure, P2P distributed feature brings higher throughput, larger scalability, and stronger robustness to P2P file-sharing systems.

2.2.2 Content Exchange Process in P2P File-Sharing Systems

In a P2P file-sharing system, in order to download desired content, each downloader must go through two necessary stages:

- locating peers with sharable content;
- connecting the located peer to download content.

Correspondingly, each uploader also needs two steps for completing content exchange:

- publishing shareable content;
- selecting downloaders to upload content.

In the following, we review these two stages respectively.

2.2.2.1 Content Information Publishing/Retrieving in P2P File-Sharing Systems

In order to help uploaders publish content information and downloaders retrieve content information, the corresponding peers in the system must first be discovered. P2P file-sharing systems build unstructured or structured logical networks mentioned in Section 2.1 to facilitate this process, i.e., using the index server method, the flood-query method, or the DHT method to discover peers.

- Index server method: This method can be used for both content publishing and content retrieving processes. With the index server method, a certain number of nodes, as servers, are popularly known by peers. To let peers know the information of servers, this information is usually published out-of-bound, like published on famous websites. Instead of directly uploading the whole content into these servers, a peer publishes its shareable content location information (e.g. this peer's location information) into the servers.

As a result, other peers can retrieve location information of their desired content from these index servers, and then directly download the content from sharing peers. The disadvantage of this index server method is obvious because it breaks the fully distributed structure of P2P systems by introducing central nodes. The failures of these central index servers will greatly affect functionality and robustness of the system. Currently, popular P2P file-sharing systems like BitTorrent and aMule/eMule employ an index server method as one of their content publishing/retrievaling mechanisms.

- Query-flooding method: Query-flooding method, which was adopted by the Gnutella P2P file-sharing system [6], helps P2P file-sharing systems maintain their fully distributed feature. It is only used to assist peers to look for content. When a peer wants to retrieve particular content information, the peer sends requests to its known peers. If these known peers do not hold the required content, the original request will be forwarded to the other peers, who are known by these known peers. Under this strategy, each peer is required to either answer the request, if it holds the content, or forward the request to its known peers. So, the request has been iteratively flooded within the P2P network. Compared with the index server method, the content information does not need to be published anymore by using the query-flooding method. However, since flooding requests cannot be infinite, the maximum hops of requests are usually limited [6], and peers could face the risk of not finding the content, even though the content really exists in the system.

- DHT method: The DHT method also maintains the fully distributed feature of P2P systems. Like the index server method, it can also be used for both content publishing and content retrieving processes. In a P2P file-sharing network using DHT technology, each peer holds a unique identity by applying a special hash function. When a peer plans to publish some content information, a hash value has first been obtained by hashing the

information with the same hash function. After that, the hashed content information is stored onto several peers whose identities are close to the hash value. The reason for storing onto several peers instead of one peer is to improve system redundancy and robustness. If a peer later wants to retrieve this published information, this peer can also first obtain the same hash value by hashing the desirable content information. And then, it can search different parts of the entire distributed hash table to locate peers who have the closer identities with the hash value, and those peers should be the peers holding the published information. As each peer must keep its neighbors' information in the DHT network, maintaining the information of neighbors can utilize a logical structure of the DHT can as a tree, a circle, a chain, etc. With this logical structure, peers can iteratively send requests to neighbors to locate other peers in the system in a relatively short period time. Currently, aMule/eMule builds a DHT called the KAD network for facilitating the content publishing and retrieving processes. BitTorrent also began to combine the DHT functionality into its network.

2.2.2.2 Content Downloading/Uploading in P2P File-Sharing System

After obtaining information of peers who share the desired content, downloaders will build connections with those peers and begin the downloading process. They can choose to download content from one single peer, and this may require a relatively long time period, if the size of content is large. In order to speed up the download process, they can also choose to download content from multiple peers simultaneously, and this is generally adopted by modern P2P file-sharing systems. To implement this multiple-downloading scheme, the content is usually separated into multiple same-size parts called chunks. If the downloading content has multiple chunks, instead of fully downloading it from one peer, the downloader can simultaneously download different chunks from different peers. Furthermore, with the chunk-based method, each downloaded chunk can be uploaded to other

peers immediately without waiting to obtain the entire file. After downloading different chunks from different peers, the downloader will reorganize those chunks, and the whole content has been recovered to itsoriginal format.

An uploader also needs to make a decision about how to allocate uploading bandwidth to downloaders. The uploader can put downloaders into its waiting queue and assign the entire upload bandwidth to a downloader selected by a particular policy like first in first out (FIFO) or round robin. The uploader may also separate its whole uploading bandwidth into multiple slots and upload to multiple downloaders at the same time.

2.2.3 Popular P2P File-Sharing Systems

In the following subsection, we introduce the currently most popular P2P file-sharing systems: aMule/eMule and BitTorrent.

2.2.3.1 aMule/eMule P2P File-Sharing System

The aMule/eMule P2P file-sharing system is one of the most popular P2P file-sharing systems with around one million online users [25]. It is composed of the aMule application [15], the eMule application [16], and the eDonkey system [26]. aMule is a Linux-based P2P file-sharing application, and eMule is a Windows-based one. Edonkey is a network, used by both aMule and eMule, providing the content publishing/retrieving service from the index servers. We know that in a general P2P file-sharing system, when a peer performs a content exchange, this peer must first collect the information of content sources. In aMule/eMule, several different source searching approaches are employed.

- ED2K method: the eDdonkey network is composed of index servers call ED2K servers. These servers do not hold the real content for sharing. Instead, the sources' information (sharing content location information) is published on to these ED2K servers by peers who share content in the network. As a result, the information of those sources can be easily retrieved from

ED2K servers by any peer. aMule and eMule implement the full functionality of the ED2K protocol, which specifies the communication between the ED2K servers and the P2P file-sharing clients. Thus, each peer running aMule/eMule application can easily search for the information of sources from those ED2K servers.

- KAD method: with the implementation of the DHT function, peers running the aMule or eMule application also build a DHT network called the KAD network. In the KAD network, content information can be published and retrieved with the help of both publishing peers and published peers. Publishing peers are the source peers with shareable content, and they publish the content information onto other peers in the KAD network; published peers are those peers in charge of maintaining the published information. In the KAD network, each peer needs to maintain its own routing table and publishing table. Each publishing peer uses its routing table to discover the corresponding published peers, and then inserts the content location information into the publishing tables of these published peers. With this publishing/retrieval methodology, each peer can also use its routing table to discover published peers and then retrieve needed information from their publishing tables.

- Source-Exchange method: this method can aid peers obtain more sources. It assumes that a peer has already obtained some sources from other sources searching methods like the ED2K mehtod or the KAD method. During the content exchange process, this peer will continuously ask for more sources from these already known sources with the Source-Exchange method.

- Passive method: the passive method is an auxiliary method used to obtain more possible sources. In aMule/eMule, each peer must maintain its knowledge about the whole network. That is, the peer must connect to some other online peers and make them as its neighbors. The passive method exploits

this pattern and prescribes that, when building connection with those online peers, each peer will ask for the connectors whether they can be the sources of its desired content.

When exchanging real content, peers in aMule/eMule also adopt a chunk-based content uploading/downloading approach. Each piece of shared content is divided into equal-sized pieces call chunks (by default the size of each chunk is 9.28MB). So, both the download capacity and uploading capacity of aMule/eMule can be improved by this chunk-based exchange process.

2.2.3.2 BitTorrent P2P File-Sharing System

BitTorrent is also one of the most popular P2P file-sharing systems in the real world with millions of online users [17]. Same as aMule/eMule, the exchanged file in BitTorrents is also divided into multiple same-size chunks (the size of a chunk is 256KB by default). Peers downloading the same file are grouped together. Thus, one grouped peer can exchange desired chunks with other peers in the same group. Unlike aMule/eMule where sharing content information is published into its own P2P network, in BitTorrent content information is published from out-of-band. To implement the out-of-band publication, a .torrent file related to the published content must be first created. This file specifies the information about the communication information of the tracker (the tracker is very similar to the index server, but it only services for a specific group), the meta data of the shared file, such as file name and file length, and the hash value of each chunk. This .torrent file is not distributed among BitTorrent's network, but it is published onto some particular websites or online forums, where users can retrieve the .torrent file and pick up the information of the tracker. The out-of-band publishing strategy simplifies the publishing process in BitTorrent. It also helps BitTorrent avoid some copyright issues.

After obtaining information of the tracker, the new attended peer will register itself onto the tracker. In addition, during the content exchange process, each reg-

istered peer will also periodically update information of their owned chunks onto the tracker. As the tracker holds the detailed information about which peer holds which chunks, each downloader can periodically retrieve information of other registered peers from the tracker and use these located peers as its sources candidates.

2.2.4 Fairness Issue in P2P File-Sharing Systems

Maintaining fairness is necessary in a P2P file-sharing system due to the existence of selfish behavior of some peers. In a traditional client-server system, each user can be safely assumed to be obedient; otherwise the user cannot obtain the provided service. However, for a typical P2P file-sharing system, in order to keep the properties of high throughput, large scalability, and strong robustness, the fully-distributed and self-organized infrastructure must be adopted to attract as many peers as possible. That is, peers are allowed to own their identities without any cost and to attend and leave the system freely. Therefore, the environment of P2P file-sharing is highly dynamic and hardly manageable. As a result, exploiting the difficulty of management in such a P2P environment, some peers may attempt to only consume the uploading bandwidth of other peers in the absence of contributing to others. This kind of peer is called a free-rider, and a large number of free-riders were discovered in P2P file-sharing systems lacking the capability to maintain fairness [2]. Obviously, the free-riding behaviors not only contribute nothing to system performance, but also brings additional workload to contributors and eventually degrades system performance. Since system performance of P2P file-sharing networks relies on each peer's cooperation, it is vital to alleviate free-riding behavior through fairness policies.

However, maintaining fairness effectively in a P2P file-sharing system is challenging. Maintaining fairness is restricted by the decentralized infrastructure and self-organization of P2P file-sharing systems. To maintain fairness, it is impractical to give up the intuitive distributed infrastructure by introducing central control nodes, and it is also hard to implement a mandatory or monetary payment-based

fairness schedule due to the high dynamic nature of peers. Thus, reciprocity-based incentive policies are usually adopted, in which a peer prefers to promote more downloading bandwidth or higher downloading priority to peers from whom it formerly obtained benefits. Under this kind of incentive mechanism, a peer can choose their partners for content exchange according to the each partner's individual previous behavior. Generous partners obtain higher priority and larger bandwidth than free-riding partners. Depending on how to obtain the information of previous behavior, the reciprocity-based incentive mechanisms can be subdivided into indirect reciprocity-based incentive mechanisms (IRMs) and direct reciprocity-based incentive mechanisms (DRMs). Under the IRMs, a global reputation level for each peer is normally calculated and distributed. Consequently the service level of peer A to peer B could depend on the former service level of B to other peers. Under the DRMs, each peer merely maintains the contribution information of their former partners. This results in the service level of A to B depending only on B's previous service level to A. IRMs are widely discussed in [27, 12, 11, 28], but they are complicated and face some implementation issues for the current P2P file-sharing systems, such as how to sychronize information among a large number of peers within a short time period. On the other hand, due to the simplicity of DRMs, they are popularly adopted in P2P file-sharing networks such as BitTorrent and aMule/eMule.

In the following, we will briefly introduce the fairness policies in BitTorrent and aMule/eMule.

2.2.4.1 BitTorrent's Fairness Policy

As one of the most popular P2P file-sharing systems, BitTorrent adopts a TFT incentive mechanism aiming to reward generous peers and penalize free-riders in exchanging a single file. Using TFT incentive policy, an uploader will divide its entire uploading bandwidth into several equal slots (by default, the number of slots is 4 according to the official BitTorrent specification). Each slot will be as-

signed to a downloader candidate from whom the uploader can obtain data with the highest downloading rate. Each upload will lasts for 10 seconds, and then the candidates will be re-selected according to their latest measured uploading rates to the uploader. Additionally, every 30 seconds the uploader runs an optimistic unchoking strategy [17] to replace one of the current downloaders with a randomly selected downloader candidate, no matter whether or not this candidate has exchanged content with the uploader earlier. The goal of this unchoking strategy is to discover better potential partners and promote newcomers.

Even though the TFT policy tries to maintain fairness, it was shown not to be effective. According to recent researches [18, 19], Free-riders could survive in BitTorrent, and they could even obtain higher download rates than TFT compliant clients. This is mainly because free-riders are able to retrieve as many as possible sources from the tracker. After that, by exploiting the optimistic unchoking strategy, they can obtain enough downloading bandwidth from these sources.

2.2.4.2 aMule/eMule 's Fairness Policy

As another very popular P2P file-sharing system, aMule/eMule attempts to maintain its fairness with a local credit system [16]. The design goal of the credit system is to reward sharing behavior. Specifically, if peer A uploaded some content to peer B, peer B would assign some credit to peer A. When peer A later wants to download desired content from peer B, using the assigned credit, peer A will be given higher priority service from peer B than B's other partners having less credit. As the content exchange process in aMule/eMule is a chunk-based process, the credit operation can also be performed during a single chunk exchange. Thus, peers with only one chunk can still earn credit from their service recipients, and peers can use credits for downloading one chunk. In addition, unlike the TFT policy in BitTorrent which can only be applied on a single file exchange process, the credit system in aMule/eMule tries to deal with multiple file-exchange situations. To implement this goal, the credit system regulates that the unconsumed

credits can reside with peers for several months [16], so peers can use the credits when they download multiple files in the future. Another noticeable feature of this credit system is that credit operations only exist between the service provider and its direct service recipients. In other words, the credit information exists locally and is not spread among other peers in the system.

The credit system in aMule/eMule also has its weaknesses. For example, the efficiency of the credit system may be low, as the credit information is not spread among all the peers. There is no explicit fairness policy to deal with free-riding behavior, so the free-riders may easily obtain benefits. These kinds of issues were approved by several research works [20, 21], which show that free-riding behavior is popular in aMule/eMule and free-riders can obtain high downloading rates.

Chapter 3

Lookup Performance Deficiencies in the KAD Network

In this chapter, we present the impact of a poorly designed incentive fairness policy on content information lookup performance. We run a wide-range measurement on the KAD network [29], which adopts a DHT technology and combines itself into aMule/eMule P2P file-sharing network. The KAD network, as one of the largest DHTs in the real world, is designed to help peers publish and retrieve sharing content information with the routing table and publishing table technology. Our measurements show that even though the routing table is well maintained, the current refresh scheme of the publishing table and the lack of the effective incentive policy cause lookup performance deficiency. To mitigate this problem, we propose three different modifications and analyze their advantages and weaknesses in the end.

3.1 Introduction

To publish and retrieve content efficiently in P2P file-sharing systems, structured methods based on a DHT technology has been widely proposed [23, 30, 24, 31]. In reality, there is one DHT-based network being widely deployed in aMule/eMule P2P file-sharing systems [32, 16] – the KAD network. According to the work of

Steiner et al. [33], the size of the KAD network is over one million online peers. Since both aMule and eMule have implemented the full functionality of the KAD network, each peer in aMule/eMule P2P file-sharing system is a peer in the KAD network by default. Consequently, the credit system in aMule/eMule is also responsible for maintaining the fairness of the KAD network. In the KAD network, publishing and retrieving content information needs the help of publishing peers and published peers. Publishing peers are the peers with shareable content who publish the content information on to particular peers, and published peers are those particular peers in charge of maintaining the published information. Each publishing peer (the uploader) uses its routing table to discover the corresponding published peers, and then inserts the content information (e.g., the content location, name, size, etc.), into the publishing tables of these published peers. Each downloader can also use its routing table to discover published peers, and then retrieve needed information from their publishing tables.

In this chapter, we thoroughly study the lookup performance of the KAD network by running multiple measurement tests in real world. Previous measurement studies [19, 25, 34, 35] usually tested the KAD network from the perspective of a single client. Moreover, due to the huge size of the KAD network, they usually measured a specific aspect of the whole KAD network. In contrast, we measure the whole KAD network through using multiple test nodes. During the measurements, the routing tables of around 20,000 peers are crawled and analyzed. More than 3,000,000 pieces of source location information from the publishing tables of multiple peers are retrieved and contacted. Based on these measurements, we have the following contributions.

1. We show that the lookup efficiency of the KAD network is low. In contrast to other content searching mechanisms (e.g., the Source-Exchange method and the Passive method) employed in aMule/eMule, little useful content location information can be retrieved from the KAD network.

2. We develop a distributed measurement framework which employs multiple

test nodes running on the PlanetLab testbed [22]. The entire identity (ID) space of the KAD network is uniformly separated into multiple parts, each of which are measured by an individual PlanetLab test node. Therefore, different from previous works, our tests provide more reliable results because of the measurement of the entire KAD network.

3. We show that the maintenance policy of the routing table is well designed. The availability of the routing table is high. That is, more than 80% of the entries in this table are connectable. Furthermore, the entries of routing tables among peers, who are logically close to each other, are different, and this causes these routing tables to have low similarity or large diversity.

4. We discover that the maintenance policy of the source-location-information publishing table is not well designed. The availability of the publishing table is low. On average, more than 75% entries in this table are stale and cannot be connected.

5. We also reveal that more than 75% peers leave the system within one hour after publishing their downloaded content into the KAD network.

6. By exploring the implementation of the KAD network, we conclude that both the current maintenance schedule for the publishing table and the poor incentive policy (the credit system) on publishing peers eventually result in the low availability of the publishing tables, which accordingly cause poor lookup performance of the KAD network.

7. We propose three possible solutions to address these issues: the self-maintenance scheme with short period renewal interval, the chunk-based publishing/retrieving scheme, and the fairness scheme. The strengths and weaknesses of these solutions are also discussed.

To improve the performance of KAD network, previous works [34, 36] mainly focused on how to deal with published peers more efficiently. However, to our

knowledge, the impact of the incentive policy on publishing peers has not been analyzed until this work.

3.2 Background of The KAD Network

In this section, we introduce the related background knowledge on the KAD network. The KAD network is a DHT network based on the Kademlia algorithm [24]. With the Kademlia algorithm, peers in the aMule/eMule file-sharing network cooperate together to build a structured overlay network for publishing and retrieving content information. In the KAD network, peers choose their neighbors according to the KAD logical distance. Each peer maintains the information of neighbors by its own routing table. To help peers locate shareable content from the KAD network, uploaders publish the information of their shareable content on to publishing tables, and downloaders can also retrieve this information from the publishing tables.

3.2.1 KAD Logical Distances

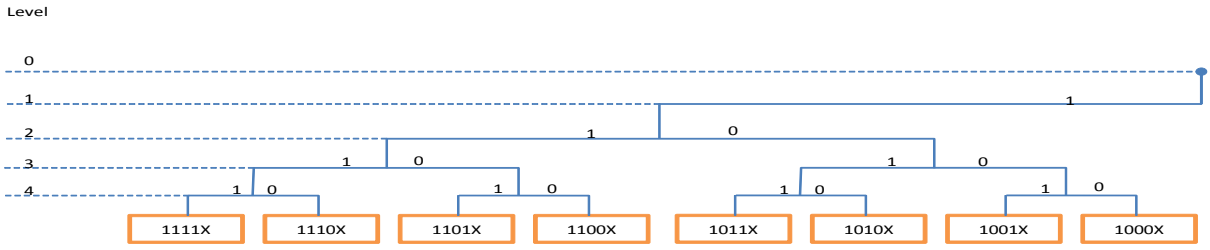
To recognize distinctive peers in the KAD network, each peer is assigned a unique identity, which is called the KAD Identity (KID). By default, the KID of each peer is generated by the peer itself using a specific hash function when it first joins the system. Since the KID is 128-bit long, the whole KID name space can theoretically cover a total number of 2^{128} different peers. Following the Kademlia algorithm, the logical distances among peers can be calculated by bitwise Exclusive-OR (XOR) operation on their KIDs. A larger value of XOR result represents a longer logical distance when compared to a smaller value of XOR result. For example, considering a 4-bit long KAD name space where peer A, B, and C's KIDs are 1010, 0101, and 1100 respectively. With the bitwise XOR operation, the logical distances between A and B is 1111 and between A and C is 0110. Consequently, C is recognized logically closer to A than B under this KID name space.

Each published item in the KAD network is also assigned a 128-bit long identity. Thus, the logical distance operations are also used to decide where the publishing items will be published and then be retrieved. Like the KID of each peer, the identity of a published item can be created by hashing its meta data, such as the content name, the content type, etc. We will discuss these publishing and retrieving processes in detail in the following subsection.

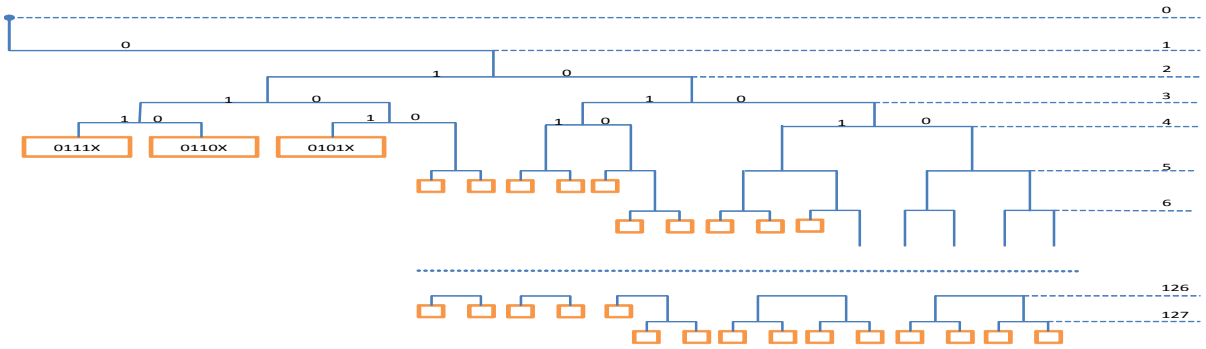
3.2.2 Routing Table and Publishing Tables

Since the KAD network is fully distributed and self-managing, each peer in the KAD network individually maintains a routing table for its knowledge on other peers. The entries in the table are the connecting information of its known peers (the neighbors), such as KIDs of these peers, these known peers' IP addresses, UDP ports, TCP ports, etc. The logical structure of a peer's routing table is represented by a binary tree as shown in Figure 3.2.1. Each level of the tree corresponds to one bit of the KID. Theoretically, the tree's height can be extended to 128 levels corresponding to the total length of the KID. Connecting information of peers is collected into the leaf nodes of the tree, and these leaf nodes are named buckets. The KIDs of the peers belonging to a bucket at a specific level will have the same prefix bits until that level. For instance, the KIDs of peers within a same bucket in level 10 have the same 10 prefix bits. A bucket may hold at most 10 entries, beyond which the bucket must be split into two buckets in the next level for holding more information. Thus, for a bucket in level 10, if more than 10 peers belonging to this bucket are known, the level 10 bucket will be split into 2 buckets in level 11.

The position of known peers are not directly assigned into the binary tree according to their KIDs. Instead, each entry's position in the binary tree is decided by its XOR distance from the routing table's owner. That is, peers, belonging to the left branch at a specific level of the binary tree, have a different corresponding bit from the routing table's owner, while peers on the right branch have the same



(a) left binary tree



(b) right binary tree

Figure 3.2.1: Logical structure of the routing table. X represents the rest bits

corresponding bit. In other words, the closer to the most left part a peer's position in the tree is, the further from the routing table owner this peer is, while the closer to the most right part a peer's position in the tree is, the closer to the routing table owner this peer is. For instance, considering the above example: peer A is the routing table's owner, and a bucket of its routing table in level 10 is split into 2 buckets in level 11. The bucket in the left side of the level 11 holds peers whose 11th bit of KID is different from the 11th bit of A's KID, and the other bucket in the right side holds peers whose 11th bit of KID is the same as the 11th bit of A's KID. However, peers in both buckets still have the same 10 prefix bits.

It is impractical and unnecessary for each peer to maintain the information of all other peers within its routing table. Otherwise, the size of the routing table will be extremely large due to millions of online peers in the KAD network, and the routing table must be refreshed frequently because of the high dynamic behavior of peers. In reality, each peer in the KAD network is only required to maintain information from more peers with a closer match of KIDs to its KID and maintain

fewer peers whose KIDs are farther away. With this regulation, the binary tree – the logical structure of the routing table – becomes unbalanced. That is, the number of leaf nodes on the left-hand branch of the tree is far less than the number of leaf nodes on the right-hand branch. In addition, there are no buckets in the levels from 0 to 3. Otherwise, these buckets must be split soon with the growing knowledge of the entire network. The left-hand branch of the whole tree shown in Figure 3.2.1(a), which holds peers that are logically far from the routing table's owner, always stops at level 4. This regulation leads the left-hand branch of the whole tree to hold a total of 8 buckets and a total amount of 80 peers at most (the first 4 bits of the XOR distances from the owner of the routing table are also shown in Figure 3.2.1(a)). These 8 buckets cannot be split into the next level. For the right-hand branch shown in Figure 3.2.1(b), there are 3 buckets in level 4. Beginning from level 5, the 5 left-most buckets in each level cannot be split anymore; while each of the other 5 right-most buckets can be split into the next level, if more than 10 peers in its range are known.

We know that there are around one million online users in the KAD network, and the number can be roughly represented by 2^20. Thus, a binary tree with a total of 20 levels can hold all of the peers within the KAD network. When applying the 20 levels to this particular structure of the binary tree, we can roughly calculate the total number of peers $n_p$that can actually be held in a routing table with the following equation:

$$n_p = (8 + 3 + 5 \times 15 + 10) \times 10 = 960. \tag{3.2.1}$$

The formula within the braces calculate the total number of buckets, where the number $8 + 3$ represents the total buckets in the first to 4th levels. 5×15 represents that there are 5 buckets in each of the following 5th to 19th levels, and the last 20th level has 10 buckets and is represented by the number 10. The

other multiplier 10 represents the maximum number of peers in each bucket. As a result, each peer's routing table keeps the information of less than 1000 peers.

With the help of this special unbalanced tree structure of the routing table, a peer can easily locate the connecting information of any peer, even though this information does not exist in its current routing table. By using the routing table, the lookup process for a particular peer can be described as follows.

- When peer A wants to connect to another online peer B, A first calculates its logical distance from B, and then looks up the corresponding bucket from its routing table. If B is close to A, the probability of finding B is high since A knows more peers around itself; otherwise, the probability of finding B is low.

- If peer B is not found from peer A's own routing table, A will do the following. A searches for peers from its routing table who have the longest prefix-matching bits with B, and then sends lookup requests to these peers. These peers receiving the requests will run the same lookup process. They either find the information of B in their routing tables and reply this information back to A, or they retrieve additional peers much closer to B from their routing tables and reply them back to A.

- When peer A receives those closer peers, iteratively, A sends requests to those peers again until A can obtain the connecting information of B.

Maintaining the routing table includes adding new peers and removing stale peers. Each peer can obtain the information of new peers, either when these new peers directly send requests to it; or when the peer requires its known peers to send the information of more peers back to it and some of sent peers are unknown to it. On the other hand, by periodically verifying the online status of each peer, the information of stale peers in the routing table can also be removed.

3.2.3 Publishing and Retrieving Processes

Publishing the entire content into other peers could bring a large amount of network traffic into the KAD network. Therefore, uploaders only publish the related content location information into the KAD network, and consequently downloaders only retrieve this information from the KAD network. As mentioned in previous subsection, each content location information has a 128-bit long identity. With the hash table technique, the location information is the value, and the identity is used as the corresponding key. When a peer wants to share some content, both the content location information (e.g., the peer's IP address, its TCP and UDP ports) and its corresponding identity are published onto peers from the KAD network. However, instead of publishing the information on to randomly selected peers, this information will be published on to some particular online peers whose KIDs have enough prefix-matching bits with the identity of the publishing information (16 bits by default). Obviously, this publication approach also simplifies the lookup process of the KAD network, because in order to retrieve a needed content, downloaders just need to search for the peers whose KIDs are close enough to the identity of the content information.

During the publishing or retrieving process in the KAD network, each peer may act two different roles: a publishing peer or a published peer.

- A publishing peer is the peer who plans to publish its shareable content information onto the KAD network. So a publishing peer normally is an uploader of a P2P file-sharing system.

- A published peer is the peer who keeps the published content information for future retrieval by other peers, and the published peer can be either an uploader or a downloader in a P2P file-sharing system.

In order to serve the publishing/retrieving process, each peer, as both a potentially published and publishing peer in the KAD network, must maintain its routing table and publishing table. Since in theory any online peer can be found through

the routing table, the routing table is also used to locate the exact published peers. After the published peers have been found, the exact information about how to find the real content location will be retrieved from their publishing tables. The processes of publishing and retrieving content information are shown in Figure 3.2.2. The dotted lines shown in Figure 3.2.2 represent the publishing process.

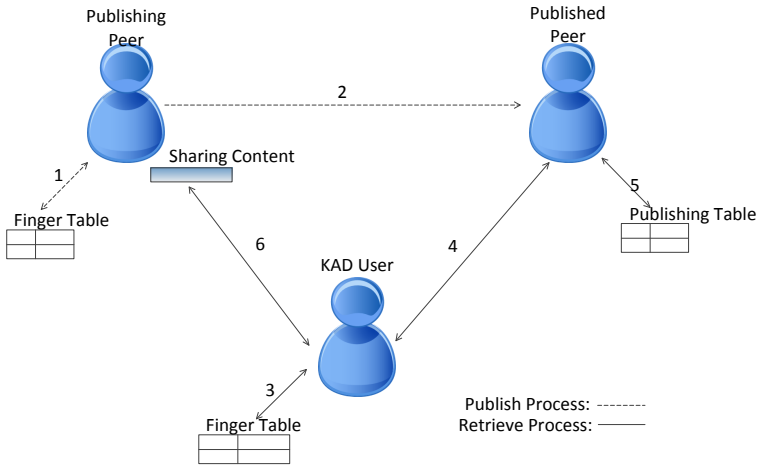


Figure 3.2.2: Publishing and retrieval processes

- When a peer publishes its shareable content location information with an identity k, this peer will select several published peers in its routing table or iteratively from routing tables of other peers whose KIDs have enough prefix-matching bits with k. The set of these peers is defined as a tolerance zone of k (step 1).

- After that, the peer inserts the identity k and its location information (e.g., its own IP address, transmission ports, etc.) into the publishing tables of the published peers (step 2).

The solid lines shown in Figure 3.2.2 represent the retrieving process.

- When a user wants to download the same content, equivalently, this user will either look up its own routing table or iteratively request other known peers to obtain the information of corresponding published peers within the same tolerance zone of k (step 3).

- After that, the user sends requests to these corresponding peers (step 4).
- These corresponding peers will search their local publishing tables, retrieve the content location information, and reply them back to the user (step 5).
- With the received location information, the user can finally connect to the publishing peers and conduct a downloading process (step 6).

3.3 Related Work

The KAD network is one of the largest deployed DHT networks integrated into the aMule/eMule P2P file-sharing system [15, 16] with millions of users [33]. Its implementation is based on the Kademlia algorithm [24], which uses the binary tree as its logic structure of the routing table and uses the XOR operation to calculate the logic distance among peers. Due to its large deployment, the research community has shown great interest in the KAD network.

Brunner et al. [29] presented a detailed analysis on the implementation of the KAD network in his master thesis. This thesis introduced the communication protocol of the KAD network, the logical structure of the routing table, the lookup process, etc. All of these analyses provided researchers a deep insight into the KAD network.

Memon et al. [37] developed a measurement tool that could accurately monitor the KAD traffic. In order to measure the whole DHT network, previous measurement tools were required to insert a large number of monitor peers into the system. Thus, the proper pattern of the DHT will be significantly affected by these monitor tools. However, Memon designed a monitor application that only replied to a limited number of targeted peers, which made the monitor invisible to most peers and consequently reduced the impact on the measured system.

Steiner et al. [38] discussed the possible auxiliary usage of the KAD network. For example, as peers could join and leave the KAD network freely, a Sybil attack [39] could be easily implemented by introducing a large number of malicious peers.

Furthermore, misusing the KAD network would easily conduct a DDOS attack. On the other hand, due to the open feature of the KAD network, it was a rare research platform with millions of online nodes for studying a distributed system. The KAD network could also be used to study how to deal with a Sybil attack and a DDOS attack [40].

Steiner et al. [41] also discovered several very interested behaviors of peers in the KAD network by their measurement study. For example, there existed a heavy-tailed distribution for the session length of peers. Some sessions sustain for multiple days, while others were over within hours. The availability of peers changed day by day and hour by hour. The KIDs were changed by peers frequently. Peers were distributed among different geographical regions such as China, Europe, Brazil, etc.

Stutzbach et al. [34] developed a tool called kFetch to measure the accuracy of the routing table. kFetch randomly chose an online peer in the KAD network and crawled its entire routing table and verified the online status of each entry. However, Stutzbach's method had the disadvantage that this measurement was issued from a single node, and the whole picture of the KAD network could not be easily obtained. According to the measurement results, about 90% of entries of a routing table were fresh. After that, Stutzbach proposed to improve the lookup performance by using a parallel lookup and increasing the number of published peers.

Steiner et al. [36] first analytically modeled the content information publishing/retrieving process. They thoroughly studied the impact of some design parameters like the number of requests for published peers and the number of actual published peers. With the measurement, they evaluated the latency of the lookup process under these different parameters. In the end, the authors suggested to reduce the lookup latency by adaptively adjusting the related parameters for published peers.

Kang et al. [42] discovered the lookup issue that users can only find few

published peers in the KAD network. They measured the KAD network through a modified client and found a high similarity of the routing tables belonging to peers close to each other. However, as their measurement method was not detailed, their thinking about high similarity among routing tables causing the poor lookup performance was inconvincible.

In conclusion, previous works tried to improve the lookup performance of KAD network by dealing with published peers more efficiently. However, the usage of the publishing peers was ignored, and the impact of the incentive policies on system performance was not addressed until this work.

Compared to previous measurement studies, we believe that our work measures the whole KAD network by adapting a distributed measurement framework via the PlanetLab testbed in the first time. We are also the first to analyze the publishing tables in the KAD network and to reveal the key factors that affect the KAD lookup performance: the current maintenance scheme for publishing tables and the lack of an effecitve incentive policy on publishing peers.

3.4 Measurement-Based Analysis

In this section, we first introduce our work that measures the lookup performance of the KAD network from a user's perspective. Then, we investigate the maintenance of both the routing table and the publishing table with real world measurements.

3.4.1 Lookup Performance of the KAD Network

To explore the lookup performance of the KAD network, we first conduct a measurement from the client's viewpoint in the real world. In aMule/eMule, each peer searches sources information with 4 different methods: the ED2K method, the KAD method, the Source-Exchange method, and the Passive method. We try to measure the efficiency of searching sources' information from the KAD network

by comparing it with other three methods.

We chose a popular P2P client application–eMule v0.49c and install it on a typical PC with a 100Mb Internet connection. This peer was required to run as a general peer aiming to download its desired content. Of course, it uploaded its obtained chunks at the same time by following the content exchange rule. To download each piece of desired content, this peer joined aMule/eMule network and looked up related sources by these 4 sources information searching methods. the ED2K method only provides sources at the beginning of search, while the KAD method, the Source-Exchange method and the Passive method can discover additional sources during the downloading process. We let this peer complete the whole downloading process for each content. After downloading one piece of content, the peer left the system, chose another piece of desired content and ran the downloading process again. During each downloading, we recorded the number of useful sources coming from different source information searching methods. In addition, we selected the downloaded content from a large range of popular, but non-copyright types like audio, video, Linux ISO-distributions, etc. The size of the content also varied from several MBs to thousands of MBs. This measurement took place over more than two months beginning at Aug. 2009. During the entire measurement, this peer downloaded over 100GB of content.

Our measurement result is shown in Figure 3.4.1. For each downloading of content, our tested peer receives 267 useful sources on average. Due to the copyright issue, the current client application does not publish information on to the ED2K servers. Thus, the ED2K method provides few useful sources. However, while the Source-Exchange method and the Passive method together provide 95% of useful sources, the KAD network only provides much fewer sources' information in comparison. As the KAD network is based on DHT technology that has been approved with high usability, apparently, we must ask what the reasons are for this low lookup performance of the KAD network. Considering the key roles of the routing table and the publishing table on the lookup process of the KAD network,

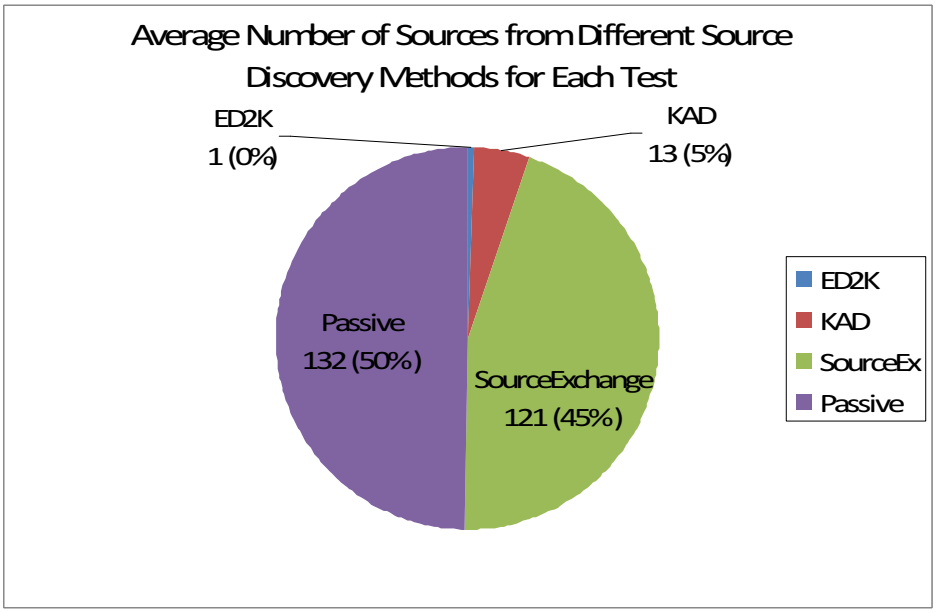


Figure 3.4.1: Received average sources information from different source-lookup methods

Table 3.1: Structure of routing table

Name	Description
Entry_ID	128-bit identity (KID) of the entry
Distance	the distance between this entry and the owner of the routing table
Entry_IP	IP address of the source
Entry_TCP	TCP port of the entry, it is used for content exchange
Entry_UDP	UDP port of the entry, it is used for KAD communication

we are motivated to explore their performance.

3.4.2 The Routing Table Measurement

The routing table is responsible for maintaining the information of online peers. The structure of a routing table is shown in Table 3.1. During publishing and retrieving processes, it is used to locate the related published peers. Thus, a poorly maintained routing table may lead to the poor lookup performance of the KAD network. For example, if the routing table is full of stale entries, uploaders may not locate enough online published peers for publishing, and downloaders may not find online published peers for retrieving. Under this kind of situation, we consider the availability of the routing table is low.

Additionally, there exists another subtle situation. When an uploader tries to

publish its content, the uploader may not be able to find the corresponding peers directly in its own routing table. Instead, the uploader will send requests to a group of peers whose KIDs are closest to the identity of the content. This group is defined as a search tolerance zone, and the default number of peers in the group is 3. The group of peers continuously searches closers in their own routing tables and may reply closers from their search tolerance zones to the uploader. Then the uploader will send requests to these replied peers. This process is iteratively sustained until the uploader finds published peers whose KIDs are close enough to the identity of the content. After that, the uploader will publish content information on to these found peers. Similarly, downloaders must locate the published peers by performing the same steps. If peers within a search tolerance zone have a large amount of same entries among their routing tables (high similarity), their returned peers to the requestors will be the same. Thus, even though many peers, whose KIDs are close enough to the identity of the content, may exist in the KAD network, uploaders/downloaders can only obtain a small part of those peers due to this high similarity. Since uploaders and downloaders maintain their routing tables independently, their finally located peers may not be the same. As a result, the downloaders cannot find the exactly published peers, and consequently the lookup performance becomes inefficient. Kang et al.[42] measured the similarity of the routing tables, and they believed that a high value of similarity exists, and this leads to the poor lookup performance of the KAD network.

3.4.2.1 Measurement Metrics

To verify these potential causes, we introduce two measurement metrics: the availability $F_a(t)$ and the similarity $F_s(t)$ of the routing table. To calculate F_a, we let n_f and n_l represent the total number of entries and the total number of the living nodes in a routing table respectively, the availability F_a of a routing table at a specific measurement time equals:

$$F_a = \frac{n_l}{n_f}. \tag{3.4.1}$$

As a result, a high value of F_a usually indicates that the entries in a routing table have high availability, and accordingly the routing table is well maintained.

Similarly, to measure the similarity F_s among m peers belonging to a specific search tolerance zone at a particular measurement time, we denote $n_f(i)$ as the total number of entries in peer i's routing table and n_s as the total number of the same items among all routing tables in all m peers. Therefore, the similarity F_s of a group of m peers can be defined as the total number of the same entries in the routing tables of these m peers divided by the average number of entries in these routing tables:

$$F_s = \frac{n_s}{\sum_{i=1}^{m} n_f(i)/m}. \tag{3.4.2}$$

Considering the impact of this similarity on lookup performance, the smaller the value of F_s, the better the routing table is maintained for lookup.

3.4.2.2 Measurement Methodology

We build a distributed measurement framework and deploy it on to multiple nodes from the PlanetLab testbed. Previous measurement studies [19, 25, 34, 35] investigated the KAD network from the viewpoint of a single client. This client was inserted into the real KAD network and either actively crawled other peers or passively monitored the communication traffic. However, due to the existence of millions of online peers, these measurements only measured a specific aspect of the whole system. Thus, their measurement results may not disclose the characteristics of the KAD network fully and accurately. In contrast, our measurement

framework can test different parts of the KAD network simultaneously. Consequently, we believe our measurement results are more accurate. We uniformly separate the entire KID name space into 16 parts. The KIDs of all peers in each part have the same most significant four bits and different 124 remaining bits, while the peers belonging to different parts have the KIDs with different most significant four bits. Each PlanetLab node is responsible for measuring a single part.

We develop two measurement applications: KADmon and RoutingTCrawl. KADmon is a customized KAD client by modifying a popular aMule client application – aMule v2.26, as our monitors. RoutingTCrawl, which is responsible for crawling the routing table of measured peers and calculating the corresponding F_a and F_s, is a crawl application developed with Python [43]. Both of these applications are installed on 16 different PlanetLab nodes. The most significant four bits, among the total 128-bit long KID of each node, are uniquely assigned from 0, 1, 2..... E, F, while the 124 remaining bits are randomly generated. Therefore, each node can measure a single part of the whole KAD network. During the measurement, each PlanetLab node joins the KAD network by running KADmon and maintains their routing table accordingly. To reduce the impact of the test on the real system, these PlanetLab monitors neither download content from any other peers nor share content with them. On the other hand, RoutingTCrawl verifies the online status of each entry from the routing table of the monitor via Ping-Pong requests (see the appendix B for detail). If the corresponding peer is online, its whole routing table is crawled, and the online status of each entry in its routing table is tested with the corresponding Ping-Pong requests.

How to crawl the routing table is minimally discussed in previous work [42, 33, 44]. However, a poor crawling method may introduce unnecessary measurement traffic and inaccurate measurement results. We present our routing table crawling method shown in Algorithm 1 in detail. In order to get more peers' information, each peer can send Kademlia-Request packets [29] to other peers in the KAD net-

Algorithm 1 Crawling routing Table

```
Data:
    list: requestList=generated requests packets
    list: targetList=peers retrieved from a routing table
    list: livingList=verified online peers
    hash(128): KID=peer's KID
    timestamp: t=timeout
    int: retrycount=# of retry when timeout
    int: parallelcount=# of requests in parallel
Initialization:
    /*according to the crawled peer's KID*/
    targetList=generateTargetKIDs( )
    /*according to the targetkidList*/
    requestList=generateRequests( )
Test:
    for each entry in requestList, do in parallel
        send request to the target peer
        if received response, then
            add to the livingList
        else if no response and timeout then
            if retrycount>0 then
                retrycount-1
                send request again
                wait for response
            else
                mark the peer offline
    /*livingList/targetList*/
    calculate availability F_a
```

work. To answer this request, the information of up to 11 peers is added into the corresponding reply message according to the KAD communication protocol. As the maximum number of entries within one bucket is 10, by carefully choosing the KID, our crawler can retrieve an entire bucket by sending one single Kademlia-Request packet. On the other hand, according to the discussion in the background section, each peer's routing table includes the information of less than 1000 peers (our measurements presented below show that on average each routing table holds the information of less than 600 peers). This also helps us design a more efficient crawler. That is, by sending less than 100 carefully created Kadmelia-Request packets in parallel, our crawler can retrieve the whole routing table of any peer within one minute. Additionally, KAD messages are transmitted by UDP packets, and both request packets and reply packets can be lost during the transmission. In order to ensure that a tested peer is offline, our crawler implements a retransmission mechanism that can filter out the lost packet situation. Moreover, by employing a multi-threading method, our crawler can test the routing tables of multiple peers at the same time. This also speeds up the measurement process and brings more accurate results when required to measure a large number of peers in a short time period.

3.4.2.3 Routing Table's Availability Measurement

We first ran our PlanetLab monitors KADmon to collect the information of enough tested peers. To obtain a unified perspective of the KAD network, this initial process on each PlanetLab node was both performed at the same time and terminated after 24 hours. After that, the RoutingTCrawl application at each monitor node was called immediately, and this application retrieved the whole routing table of the monitor and crawl each obtained peer from the table. Around six thousand peers, distributed among the entire KID name space, were crawled. Every entry of the routing table of each crawled peer was retrieved, and the online status of each retrieved peer was tested.

The measured data is shown in Table 3.2. Figure 3.4.2(a) shows the average values of availability F_a for each of these 16 different aspects of the KID name space. No significantly different values of F_a exist among these 16 parts. As the histogram and the CDF of F_a shown in Figure 3.4.2(b) and 3.4.2(c) respectively, there are more than 80% of living peers in the routing table during the measurements. With this high availability, users can find their needed online peers from the routing tables for their publication/retrieval. This high availability of the routing table benefits from the current maintaining schedule of the routing table, with which each bucket is refreshed every minute and stale peers are removed at the same time. In conclusion for this measurement, the availability of the routing table is not an issue for the lookup performance.

Table 3.2: Measurement data for routing table's availability

Data	Value
Total Number of Crawled Peers	5985
Average Number of Entries	549
Average Availability F_a	0.819
Standard Deviation of F_a	0.050

3.4.2.4 Routing Tables' Similarity Measurement

Kang et al. [42] measured the routing tables between two peers within a 16-bit search tolerance zone and indicated that those routing tables have a high similarity of 70%. They empirically selected the number 16 by considering millions of peers uniformly distributed among the KAD network, and the 16-bit search tolerance zone will cover more than 20 peers. However, only testing the similarity between two peers is not enough. In order to publish or retrieve information within the KAD network, each time, peers must conduct at least three peers in the same zone and obtain information from their routing tables [36]. Therefore, in our similarity measurement, besides testing the routing tables' similarity of two peers, we also measure the routing tables' similarity among more than two peers.

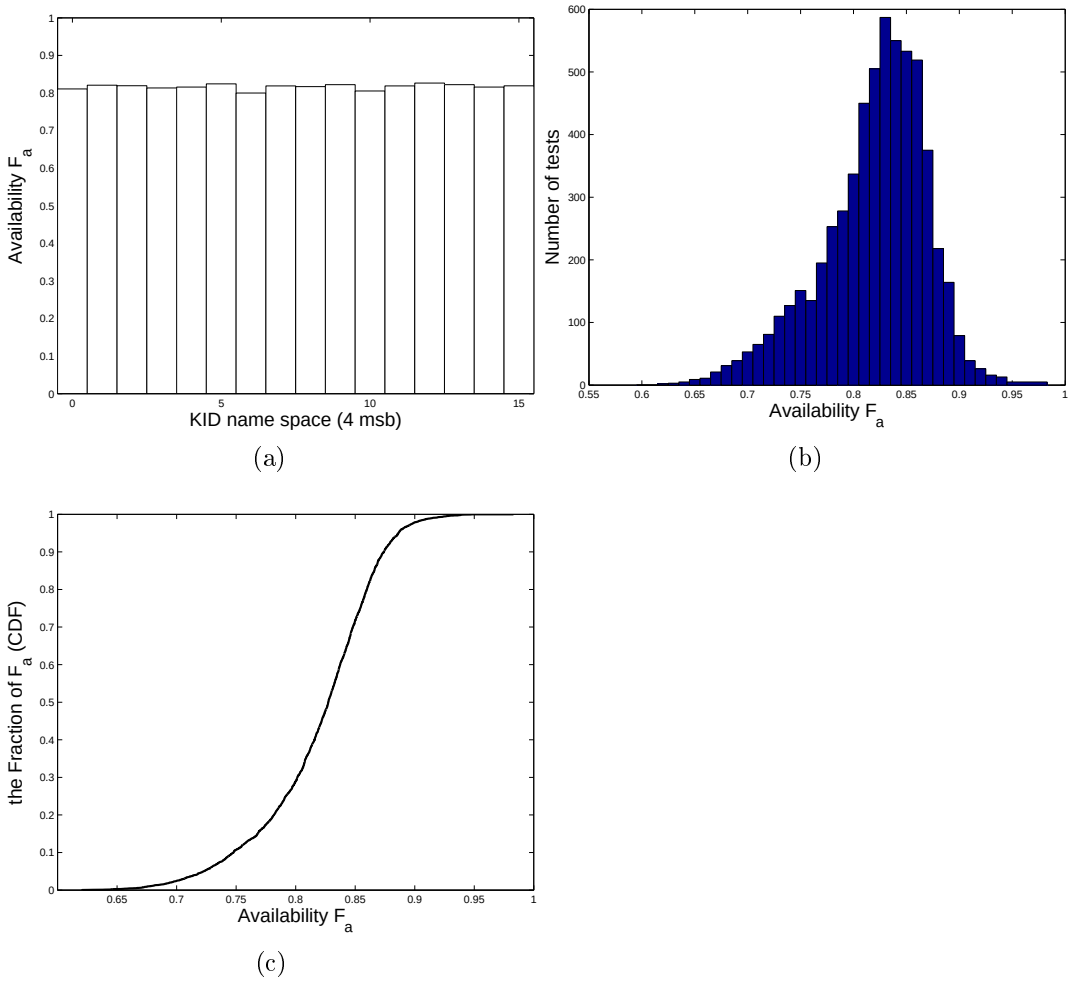


Figure 3.4.2: (a) average value of F_a measured in different aspects of the KAD name space; (b) Histogram of the peers availability F_a of the routing table; (c) CDF of the peers availability F_a of the routing table

Following Kang's work, we measured the routing tables' similarity of peers belonging to a 16-bit search tolerance zone. Within this zone, the KIDs of all peers have at least 16-bit prefix-matching. In order to obtain more accurate results, we measured different parts of the KAD network by our distributed framework. Using our measurement component KADmon, we collected a large number of peers after 48-hour monitoring. These peers were retrieved by another measurement component RoutingTCrawl, and then their routing tables were crawled. From the crawled routing tables, the peers belonging to the same 16-bit search tolerance zone were selected, and their routing tables were crawled again for calculating the similarity F_s according to Equation 3.4.2. In order to obtain more peers for

Table 3.3: Measurement data for routing tables' similarity

# of compared routing tables	# of tested tolerance zones	# of tested peers	average value of F_s	Standard Deviation of F_s
2	4120	8240	0.171	0.050
3	1018	3054	0.093	0.043
4	329	1316	0.069	0.029
5	102	510	0.060	0.022
Summary	5569	13120		

comparison, we also continuously retrieved new appropriate peers from the routing tables of these crawled peers and crawled their routing tables again. During the test, the routing tables of a total number of more than 13,000 peers belonging to more than five thousand different search tolerance zones have been crawled and compared.

The measurement results are shown in Table 3.3 and Figure 3.4.3, where the average value of the similarity F_s, the standard deviation of F_s, and the CDF of F_s among 2, 3, 4 and 5 peers within a specific tolerance zone are presented. Our measurement reveals that the similarity F_s within two individual peers is only less than 20%, which is in contrast to Kang's work. Furthermore, the similarity significantly reduces to less than 10% when comparing to more than two individual peers within a same tolerance zone. Therefore, the routing tables of the peers within a 16-bit tolerance zone have little similarity, and actually a large amount of diversity. The reason for this low similarity can be explained as follows. Under an open KAD environment, individual peers have their own individual behaviors: they may arrive and leave the system at different times, or they may meet different peers when sharing or requesting different content. Consequently, their perspective to the KAD network becomes differentiated. Moreover, the design of the KAD network may also deliberately aggravate the highly diversified viewpoint among each peer. According to the KAD rule of maintaining the routing table, when a peer attempts to fill out its routing table, it is designed to randomly search for more new peers instead of explicitly synchronizing the routing tables among

its close peers. There is a large difference between our measurement results and Kang's work. Since Kang et al. [42] did not present their measurement in detail, the accuracy of their test cannot be evaluated.

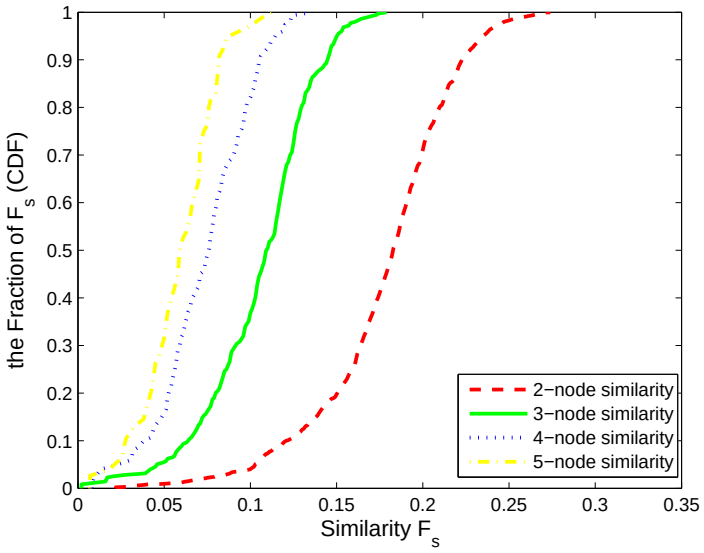


Figure 3.4.3: CDFs of similarity for different number of compared nodes

On the other hand, we believe the low similarity of routing tables does not degrade, but improves the lookup performance in the KAD network. The low similarity can keep publishing peers having a much broader perspective to the whole KAD network. As a result, more peers who belong to the same search tolerance zone can be found and chosen for the publication. This also causes peers to retrieve the published information more easily, even though some of the published peers leave the system, when peers look up all of them. In contrast, if the routing tables of the peers within a search tolerance zone have a high similarity, the information will be published onto a small portion of published peers. When this small portion leaves the system or when the left portion of peers are tried to be located, little useful information can be obtained from the KAD network. In summary, high similarity is not the critical issue for the poor lookup performance, since it is not detected.

Table 3.4: Structure of SLI publishing table

Name	Description
Content_ID	128-bit content identity
Source_ID	128-bit uploader identity (KID)
Source_IP	IP address of the source
Source_TCP	TCP port of the source, it is used for content exchange
Source_UDP	UDP port of the source, it is used for control communication

3.4.3 The Publishing Table Measurement

Poorly maintaining publishing tables will also tremendously affect the lookup performance. After peers, through their routing tables, first locate the published peers whose publishing tables are holding the desired content, peers must eventually look up uploaders from these publishing tables. If entries in the publishing tables are stale, the retrieved information will be useless.

In the KAD network, two kinds of publishing tables are used to maintain publishing information: the source-location-information (SLI) publishing tables and the keyword-information (KI) publishing tables. The main structure of these tables are shown in Table 3.4 and Table 3.5 respectively. The source location information will be inserted into SLI publishing tables of published peers. The related keywords for the publishing content will be inserted into the KI publishing tables. While the KI table conveniently helps users look for desirable content by meta data, such as file name, file type, etc., the SLI publishing table is more important because it contains the actual location information of content. In the following, we conduct real world measurements to verify the availability of the SLI publishing table. Then, we use the obtained results to evaluate the maintenance of the KI table, for the KI table is also maintained by the same schedule of the SLI table.

Following the same distributed measurement methodology in Section 3.4.2, we test the publishing tables of multiple nodes that are uniformly distributed on the entire KID name space. We still use the application KADmon as the monitor, but develop a new crawl application called PublishTCrawl to crawl publishing

Table 3.5: Structure of KI publishing table

Name	Description
Content_ID	128-bit content identity
Source_ID	128-bit uploader identity (KID)
Content_Name	the full name of the content
Content_Size	the byte size of the content
Content_Format	the type of the content (.mp3, .iso, ...)

tables. PublishTCrawl employs the same technique used by the RoutingTCrawl component in Section 3.4.2. It is also deployed on each PlanetLab monitor node. This time all 16 monitors from the PlanetLab testbed act as published peers. They accept the relevant publishing information and build their own publishing tables accordingly. Every half hour, the entire SLI publishing table of each monitor is recorded into a local log file. At the same time, PublishTCrawl is triggered. It reads this local log file and tests the online status of each entry from the SLI publishing table by sending the KAD Ping-Pong packets. Similar to the routing table measurement metric F_a, we also define a measurement metric P_a to measure the availability of the SLI publishing table. For a single SLI publishing table, P_a is equal to the total number of the information of living peers n_e divided by the total number of the information recorded peers n_r in the table.

$$P_a = \frac{n_e}{n_r}. \tag{3.4.3}$$

The measurement was run on 16 PlanetLab-based monitors for a total of 25 hours. In each half-hour trial, on average, the online status of four thousand peers retrieved from publishing tables were tested. As a result, 800 measurement trials were totally conducted, and a total number of more than 3,000,000 peers have been connected.

The measurement results shown in Figure 3.4.4 explicitly reveal that the SLI publishing table is not well maintained. We believe this poorly maintained publishing table causes the poor lookup performance in the KAD network. According

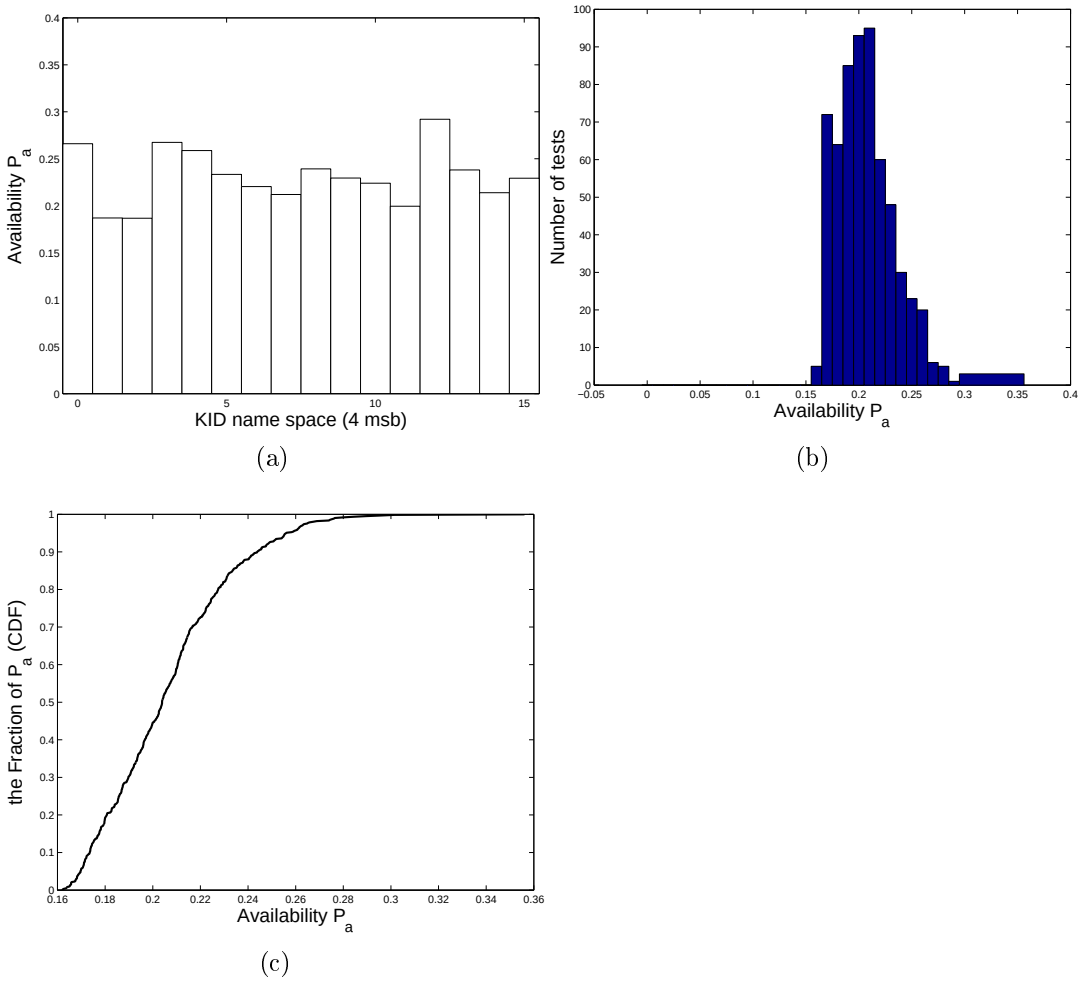


Figure 3.4.4: (a) average value of P_a measured in different aspects of the KAD space; (b) histogram of the source-location-information publishing table; (c) CDF of the availability P_a of the source-location-information publishing table

to Figure 3.4.4(b) and (c) shown, on average, only less than 25% of peers recorded in the publishing table are actually online. Consequently, when users request source location information from these SLI tables, more than 75% of the entries from the corresponding replies are useless. Additionally, the minimum value of measured P_a is even closer to 0.15, while the maximum value of measured P_a is less than 0.35. Figure 3.4.4(a) also indicates that this trend is consistent within the entire KAD name space.

Why is the publishing table not well maintained? To answer this question, we first explore the KAD maintenance algorithm for the SLI publishing table. The KAD protocol requires each publishing peer to be responsible for maintaining their

published information. According to the identity of the published information, this information is inserted into publishing tables of the published peers whose KIDs are close. In addition, when the source location information is inserted into publishing tables for the first time, a living-time period of 5 hours is also attached to this record. After 5 hours, if the corresponding publishing peers are still online, they will renew this information for another additional 5 hours. If the publishing peers are offline, their publishing information will be automatically removed out of the publishing tables after 30 minutes. Thus, even though the publishing table is held by published peers, the right of maintenance belongs to remote publishing peers. This method has no problem when the publishing peers stay online for a long period. However, if publishing peers leave the system within 5 hours after its publication, both the published location information and the published keywords information become useless when the users subsequently retrieve the information during the rest of the time period. Additionally, the KI publishing table is also maintained by the same scheme, except that the living-time period for each publishing entry is extended to 24 hours. Thus, it also suffers from the same weakness as this maintenance scheme.

An incentive policy that is called a credit system is actually employed in aMule/eMule P2P file-sharing systems [45]. This credit system aims to promote sharing and accordingly to penalize the selfish. Since a peer in the aMule/eMule is also the same peer in the KAD network, incenting peers in aMule/eMule also incents peers in the KAD network. In aMule/eMule, after downloading the whole content, the downloaders are forced to publish the content information onto the KAD network immediately (they also become uploaders). If the credit system can prolong the stay time of uploaders, they can be located through the publishing table in the future. Otherwise, if peers leave the system within 5 hours of publication, the entries from publishing tables cannot be reached any more for the rest of the time. This will be one of main issues that cause the low availability of publishing tables and consequently one of the critical reasons for the inefficient

lookup performance.

Can the credit system prolong the stay time of publishing peers for at least 5 hours? To answer this, we conduct another measurement study to examine the behavior of publishing peers. We still use our distributed measurement framework. The 16 monitors as published peers are responsible for collecting the information of publishing peers. Every 30 minutes, the crawler component PublishTCrawl retrieves each entry from the publishing table of the monitor and tests its online status. This test has been performed for 24 hours. Our measurement data shown in Figure 3.4.5 verifies that the majority of the publishing peers (more than 75% of them) leave the system within one hour after they publish their content. According to our discovery, even though a credit incentive policy has been employed by both aMule/eMule P2P file-sharing system and the KAD network, the majority of peers will still leave the system quickly after they finish their downloading tasks. Therefore, peers in aMule/eMule and the KAD network are selfish, and the current credit system is less useful.

In conclusion, the reasons for the low availability of the publishing tables and the poor lookup performance of the KAD network can be summarized as follows:

- the selfishness of the publishing peers and the poor incentive policy;
- the publishing tables of published peers are maintained by the publishing peers;
- the poorly designed 5/24 hours maintaining schedule for both the SLI publish tables and the KI publishing table.

We noticed that in previous works[36, 34, 42] the main focus is improving the performance at the side of published peers. However, with the selfish departure of the publishing peers, the publishing information becomes meaningless. Therefore, we argue that controlling behaviors of peers should become a necessary consideration for the KAD design. Actually, a more effective incentive policy should be designed to deal with the high level of selfishness.

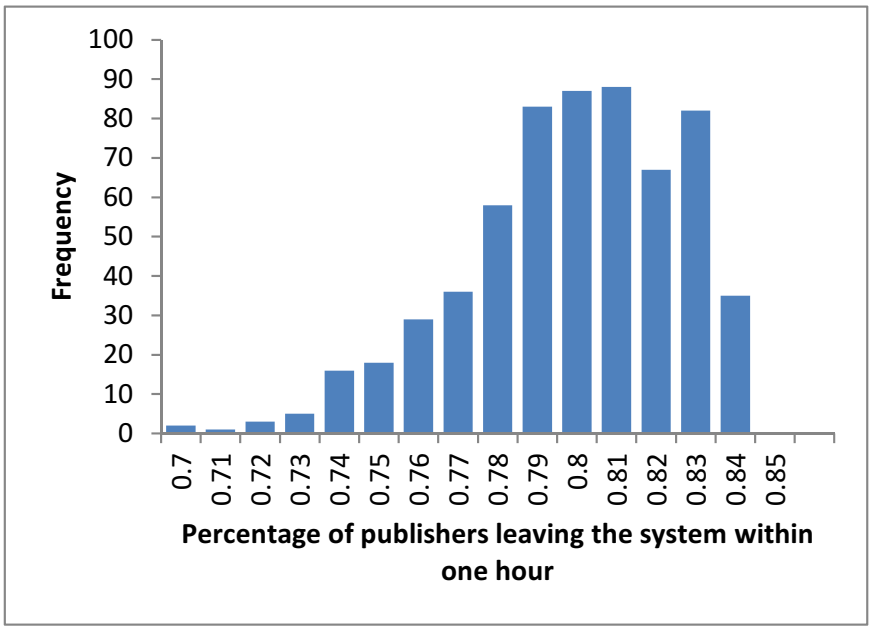


Figure 3.4.5: The histogram of the percentage of selfish peers leaving the system within one hour after publishing content to the KAD network

3.5 Possible Solutions

To solve the low lookup performance issue of the KAD network, we discuss three possible schemes in this section and show their advantages and weaknesses respectively.

3.5.1 Self-Maintenance Scheme

One possible scheme is to assign the task of maintaining publishing tables back to the local peers (published peers) themselves. The published peers can verify the online status of entries in their publishing tables within a short time period (e.g. 10 minutes). They can also directly synchronize the information of peers from their well-maintained routing tables. With this modification, stale records in the publishing tables can be significantly reduced, and retrieved information by other peers can be more useful. Other advantages of this self-maintenance scheme may include: it does not change any fundamental structure of the current system; it unifies the maintenance pattern of the routing table and the publishing tables; it may also be easily implemented. However, the self-maintenance scheme improves

the availability of the publishing tables just based on reducing the number of useless records. Because of the selfish behavior of peers and poor incentive policy, this scheme cannot fundamentally improve the lookup performance of the KAD network. The total amount of useful information available to users was not increased. Additionally, since it needs to renew publishing tables frequently, it may also introduce far more table-maintaining traffic into the system.

3.5.2 Chunk-Based Publishing/Retrieving Scheme

We can use a chunk-based publishing/retrieving scheme to replace the current file-based publishing/retrieving scheme. In aMule/eMule, the chunk-based downloading and uploading scheme has already been employed to speed up the content sharing process. Under the chunk-based scheme, a file is usually separated into several chunks. Each chunk can be downloaded from different peers and uploaded to others simultaneously. However, instead of publishing the information of an obtained chunk immediately, a peer in aMule/eMule publishes the information of an entire file into the KAD network. That is, under the current publishing/retrieving scheme, the file will not be published into the KAD network until it has been fully obtained.

In a typical P2P file-sharing environment, a peer usually runs both the uploading and the downloading processes concurrently, and this peer will not leave the system before it completes its download tasks. Thus if the information of peers, who possesses some chunks of the whole file, can also be published, the availability of the publishing table will be significantly improved. The obvious advantage of this modification is that it can increase the amount of useful information for users retrieving in the future. However, this scheme requires a redesign of the publishing/retrieving scheme, which may not be easily implemented. Moreover, it will also complicate the maintenance of the publishing tables and introduce more traffic into the system.

3.5.3 Strict Fairness Scheme

The third solution is to adapt some effective incentive policy to mitigate the selfish behavior of peers. Currently, aMule/eMule has already deployed a local credit system to reward the sharing behavior. However, this policy is neither efficient nor fair [45]. The KAD network has not employed any fairness strategy to promote the publishing and retrieving process exclusively. Since the selfish behavior of peers will lead to the low lookup performance of the KAD network, some strict fairness modification must be introduced into the KAD network. One possible strict fairness scheme may base on how to fairly allocate sources location information among requestors. After receiving requests for sources location, the KAD network will return a different number of sources according to the credit that the requester has, while the credit value depends on peers' previous sharing behavior in the KAD network. The detailed policy is as follows.

- For the requester who stays longer and shares enough content location information in the KAD network, i.e., it will be awarded a high value of credit, the requestor would be provided all known sources information for the KAD network. Through this rule, publishers may be promoted to stay in the system longer for obtaining more credit.

- For the requester whose credit value is lower than a threshold or without credit, the number of returned sources information would be proportional to the requester's credit. This rule aims to punish publishers who will leave the system soon after publishing their downloaded content into the KAD network.

With this scheme, if a publisher leaves the system soon, the publisher's benefit will be decreased when it tries to retrieve information in the future and vice versa. In addition, since adopting the fairness-based modification will keep the publishing/retrieving scheme of the KAD network unchanged, it may make the implementation relatively easy. However, even though the modification based on

Table 3.6: Comparison of different modification schemes

Methods	improve P_a	improve the amount of useful information	reduce selfish behavior	change current structure	introduce more traffic
current scheme	No	No	No	No	No
self-maintenance	Yes	No	No	No	Yes
chunk-based	Yes	Yes	No	Yes	Yes
fairness policy	Yes	Yes	Yes	Yes	Yes

the fairness design has a great advantage by fundamentally reducing the selfish behavior of peers and accordingly improve the lookup performance of the KAD network, the introduction of a strict fairness policy will still increase the complexity of the KAD network.

We have summarized the comparison of these modifications in Table 3.6. It looks like no modification is perfect. We believe that a combination of these possible modifications will be the most valid scheme to deal with the lookup performance issue of the KAD network.

3.6 Conclusion

This chapter focuses on investigating the reasons for the poor lookup performance of the KAD network. As the key components in the publishing/retrieving process, the maintenance of both the routing table and the publishing table are tested by several large-scale measurements. To keep the results accurate, the distributed measurement framework is deployed onto multiple nodes from the PlanetLab testbed. First, the availability and the similarity of peers' routing tables are tested. For the availability of the routing table, test results show that on average more than 80% of nodes in the routing table are online. For similarity, test results show that less than 25% of records are the same among different routing tables of peers belonging to a tolerance zone. Therefore, the routing table is shown to be well maintained, and it can help peers find their desired partners easily. After that, the source-location-information publishing table is tested. Test results show

that on average only around 25% of items in this table are online. Furthermore, the average staying time of the publishers is also measured, and the measurement results reveal that over 75% of publishers leave the system within one hour after publishing content. As the publishers are required to maintain its published content information, the maintenance policy regulates that their published content information will stay online for a relative long period (5 hours by default). Noticed that the current incentive policy, the credit system, cannot keep publishers staying at the KAD network for such a long time. The current maintaining method for the publishing table, the poor incentive policy, and the selfishness of the publishing peer are the reasons for the low availability of the publishing tables, which accordingly cause the poor lookup performance of the KAD network. Finally, three possible modifications are proposed to deal with this lookup performance deficiencies: a self-maintenance scheme, a chunk-based publication/retrieval scheme and a strict fairness scheme. Their advantages and drawbacks are also compared.

Chapter 4

Evaluating Stranger Policies in P2P File-Sharing Systems with Reciprocity Mechanisms

Treating strangers carelessly in a P2P file-sharing system will cause the degradation of system performance and fairness. In this chapter we evaluate the impact of different stranger policies on system performance and fairness. First, we present the necessity of designing an effective stranger policy in a typical P2P file-sharing system. Then, we review the related research work about P2P's performance and fairness. After that, in the case of the indirect reciprocity incentive mechanism being used, we adopt both numerical analyses and agent-based simulations to evaluate the impact of a broad range of stranger policies from extremely rewarding strangers to extremely restricting them. In the case of the direct reciprocity incentive mechanism being used, we conduct an agent-based simulation model and use it to reveal the impact of stranger policies on system performance and fairness. Finally, two cases studies on BitTorrent and aMule/eMule provide deep insight into the design trade-off of stranger policies.

4.1 Introduction

A typical P2P file-sharing system usually holds a large number of users and possess the features of distribution, self-organization, and self-management. P2P file-sharing systems prefer a large population because it can bring high system throughput, large scalability, and strong robustness. In order to attract as many peers as possible, P2P file sharing systems, like BitTorrent [17] and aMule/eMule [15, 16], tend to provide open environments (i.e. allowing peers to create their own identities independently and without any cost; permitting peers to attend and leave the system freely). However, even though this open environment can promote the population of peers, it may also provide additional benefits to selfish peers. The extremely selfish peers only consuming the downloading resource without any contribution, called free-riders, always easily exist in an open P2P file-sharing environment. The system performance of P2P file-sharing systems fully relies on each peer's cooperation. Obviously, free-riding behavior contributes nothing to system performance, in addition, it harms contributors and eventually degrades system performance.

To maintain system performance, the free-riding issue has to be treated seriously, and fairness, which promote generous behavior and alleviate free-riding, usually has to be maintained in P2P file-sharing systems. Otherwise, free-riding may become popular. A previous measurement study in [2] indicated that around 70% of peers in Gnutella, a famous P2P file sharing system which lacks a fairness mechanism, were free-riders. The characteristics of P2P file-sharing systems in autonomy, self-organization, and self-management give much more freedom to peers. A peer in a P2P file-sharing system is controlled by an individual person who has the ability to choose, independently, whether to contribute or not. In addition, peers may locate in different regions or belong to different service providers. They may have different uploading bandwidths and downloading bandwidths. Thus, a low contribution level may not represent selfish behavior. As a result, mandatory management in term of a uniform standard may introduce additional unfairness

to them. Consequently, a fairness policy based on incentive mechanisms is naturally the appropriate method, in which collaboration is promoted and free-riding is prevented.

Fairness in P2P file-sharing systems is usually maintained by reciprocity-based incentive mechanisms [3, 4], in which the benefit of peers is based on their historical sharing behavior. These reciprocity-based incentive mechanisms are commonly divided into two typical classes [4]: indirect reciprocity mechanisms (IRMs), which are widely discussed in [27, 12, 11, 28], and direct reciprocity mechanisms (DRMs), which are popularly adopted in P2P file-sharing networks such as BitTorrent and aMule/eMule. Under the IRMs, a global reputation level for each peer is normally calculated and distributed, and consequently, peer A's service to peer B could depend on B's former service to other peers. Under the DRMs, each peer merely maintains the information of their former partners, and this results in A's service to B depending only on B's previous service to A.

4.1.1 The Stranger Policy

To design an effective reciprocity-based incentive mechanism, we believe how to deal with strangers is indispensable. When contributors choose uploading candidates, they must make decisions on whether or not to provide content to strangers whose historical behavior information are not known. In a P2P file-sharing system, there may exist different definitions for strangers, which will affect how to design stranger policies considerably. For instance, if the strangers are recognized as peers just arriving into the system, the time factor must be included to judge strangers, such as how long a stranger will be a known peer. If the strangers are defined as peers without any sharing behavior, a strict policy should be adopted to penalize strangers. In this chapter, we define the stranger from the viewpoint of the uploaders. Considering that each uploader in a P2P file-sharing system must individually choose downloading requesters for uploading, a stranger is a candidate peer, who has been unknown by the uploader, no matter how long this candidate

peer stays in the system. The word 'unknown' here means either the uploader has not exchanged content with this stranger before, or the uploader cannot obtain the behavior information of this stranger from other peers. Corresponding with strangers (unknown peers), another kind of peer called known peers also exist in the system. For an uploader, known peers are a kind of peer who either exchanged content with this uploader, or their sharing information with other peers can be obtained by this uploader. According to an incentive policy, a known free-rider cannot be serviced by its uploader. Thus, the known peers in the following parts will refer to the known general peers, who legitimately upload and download by following P2P rules. In the following sections, the word "strangers" is used to represent the unknown peers.

The reciprocity-based incentive mechanisms work according to the historical behavior of peers. Thus, these mechanisms are helpful for managing known peers. However, the mechanisms are incapable of tackling the strangers because strangers do not have any behavior information for their uploaders. In a typical P2P file-sharing system with a reciprocity-based incentive mechanism, strangers are the following peers:

- Under the IRMs, strangers can be new arrival general peers (legitimate newcomers) or whitewashers. The general newcomers are peers who arrive into the system for the first time. They have not exchanged data to others and are considered strangers by their current partners. However, the general newcomers can potentially become known peers after uploading content to their partners. The whitewashers [9] are free-riders who pretend to be newcomers for more benefits by leaving and rejoining the system with updated new IDs. By frequently changing their IDs, whitewashers become strangers to their partners. We know that to obtain the benefits intended for strangers, free-riders in a P2P file-sharing system with the IRM have strong motivations to whitewash. Otherwise, they could be easily recognized and isolated.

- Under the DRMs, the strangers for a uploader can be divided into three

categories: general newcomers, free-riders who have not downloaded content from this uploader (otherwise, they will be recognized as free-riders by this uploader), and general peers who have not exchanged content with the uploader (but they have exchanged content with other peers). We know that free-riders do not need to whitewash to benefit under the DRMs. This is because the local knowledge of general peers is not spread to others. Therefore, free-riders do not have to worry about their selfish behavior toward one peer being known and accordingly being punished by other peers.

To help readers figure out the scopes of free-riders, strangers, and whitewashers, we illustrate them in Figure 4.1.1

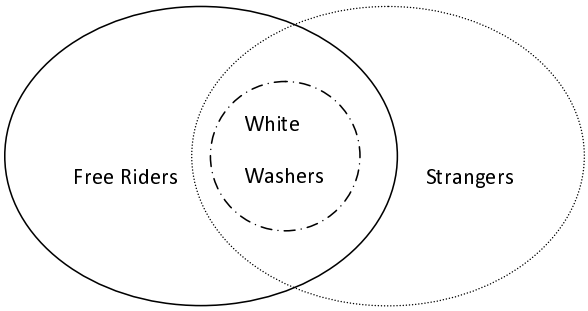


Figure 4.1.1: The relationship among free-riders, strangers, and whitwashers.

Since every uploader (the resourceable general peer) faces the issue of how to deal with such strangers efficiently in an open P2P environment, treatment policies need to be established. Even though the impacts of stranger policies to general newcomers and free-riders are respectively intuitive, the impacts of different stranger policies to system performance and fairness cannot be decided easily, because we cannot distinguish free-riders from general newcomers in a typical P2P environment [9]. For system fairness, restricting strangers policy is not necessarily better than rewarding strangers policy. The reason is that the policy of restricting strangers treats free-riders fairly, but treats general newcomers unfairly; while the policy of rewarding strangers treats general newcomers fairly, but treats free-riders

unfairly. Similarly, for performance, rewarding strangers is also not necessarily better than restricting them. As we know, promoting the potential contribution of general newcomers will also promote selfish consumption of free-riders. Thus, rewarding strangers can provide more uploading bandwidth to systems, but may also provide more bandwidth to selfish competitors at the same time. On the other hand, limiting the consumption of free-riders will unavoidably limit the contribution of general newcomers. As a result, restricting strangers reduces selfish competitors in systems, but also reduces the total uploading bandwidth. Even though the treatment policy for strangers affects system performance and fairness remarkably, few quantitative evaluations for different stranger policies currently exist in literature [3, 46].

4.1.2 Contribution

Maintaining fairness in P2P file-sharing systems normally focuses on how to fairly allocate resources (i.e. the upload bandwidth of resourceable general peers) among peers. Specifically, each uploader needs to independently make the decision of how to allocate their uploading bandwidth to others. In this chapter, we focus on an anonymous, open P2P file-sharing environment with the IRMs/DRMs, where each peer can independently assign upload bandwidth to their known peers and strangers according to an adopted stranger policy. We evaluate the impact of different stranger policies on system performance and fairness through the following methodology.

1. We define a performance metric (Equation 4.3.5) and a fairness metric (Equation 4.3.4). The performance metric is used to measure the average downloading rate of general peers in the system. The fairness metric is used to evaluate whether or not peers are treated fairly.

2. We define stranger policies as the total percentage of the uploading bandwidth of each resourceable general peer assigned to strangers. As shown in

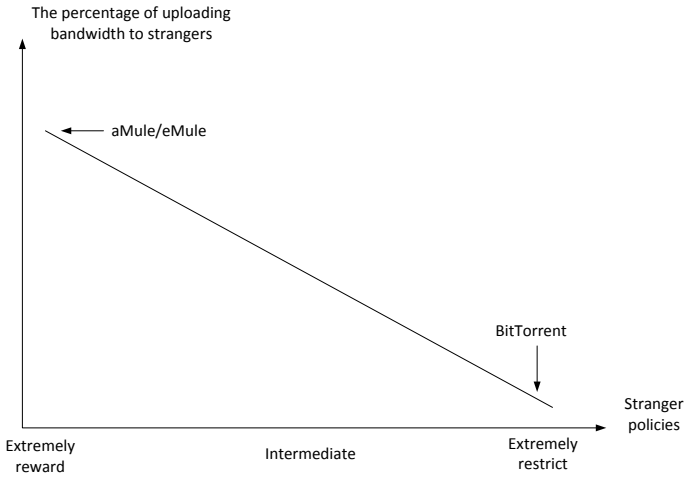


Figure 4.1.2: The relationship between the spectrum of different stranger policies and the percentage of the uploading bandwidth to strangers. The x axis represents different stranger policies, and the y axis represents the percentage of the uploading bandwidth to strangers.

Figure 4.1.2, the leftmost point represents the extremely rewarding stranger policy that gives most of the uploading bandwidth to strangers. In contrast, the rightmost point represents the extremely restricting stranger policy which assigns an extremely small percentage of the uploading bandwidth to strangers. The intermediate points represent the stranger policies giving the gradually decreasing uploading bandwidth to strangers with the direction of the arrow of x axis.

3. With the IRMs, we use both numerical analyses and agent-based simulations to evaluate the whole spectrum of these different stranger policies shown in Figure 4.1.2.

4. With the DRMs, we adopt an agent-based simulation to study the impact of these different stranger policies on system performance and fairness.

5. We illustrate the validity of this simulation model by analyzing and testing the stranger policies in BitTorrent and aMule/eMule.

Based on this methodology, the following are the key findings:

1. The extremely restricting stranger policy brings the best fairness at the cost of the degradation of performance. The extremely rewarding stranger policy cannot provide the highest performance to the system. Appropriately choosing the intermediate policy shown in Figure 4.1.2 can bring the highest performance to the system. The system performance can be improved when the potential contribution of general newcomers is quickly promoted, while free-riders could also obtain benefits at that point.

2. The results reveal that under different stranger policies, the varying tendency of performance and fairness are not consistent, and the optimal performance and fairness (free-riders will not survive) of the system cannot be reached simultaneously. Specifically, when the system reaches the best fairness where a very small fraction of uploading bandwidth is allocated to strangers, free-riders will be significantly restricted and cannot easily obtain benefits from the system. However, the system performance will be negatively affected due to the delay of potential contribution of general newcomers. When the system reaches the highest performance where at least some part of the uploading bandwidth is allocated to strangers, free-riders also receive this benefit and survive in the system. Moreover, due to free-riders surviving at this point, general peers may tend to free-riding since they are rational and receive less benefit than free-riders, or they may directly leave the system due to being treated unfairly. Eventually, these consequences will negatively affect the system performance. Taking this design dilemma into account, P2P designers are required to tackle strangers carefully according to their individual design goals.

3. Different realistic P2P file-sharing networks have different design objectives. According to the agent-based simulation, we show that BitTorrent prefers to maintain fairness by sacrificing its performance, while the fully rewarding

stranger policy of aMule/eMule promotes the benefit of free-riders. We also suggest some possible alternative improvements for both BitTorrent and aMule/eMule.

4.2 Related Work

Researches on P2P file-sharing systems have grown in the past years. Not only the performance, but accordingly the fairness and the major management approach on maintaining performance in P2P systems, have also been widely studied. This section presents the related work in studying the performance and fairness of P2P systems.

4.2.1 Research on Performance

To evaluate the performance of P2P file-sharing systems, the research community adopted different approaches, such as mathematical analyses and empirical studies.

4.2.1.1 Numerical Analyses

Typical P2P systems have tremendous population. The behavior of each peer is random. Thus, compared to other research approaches, mathematical analysis would be a fast and effective method that can capture the main performance characteristics of P2P systems. The following paragraphs summarize the relative contributions from numerical analysis of P2P performance.

Z. Ge et al. [47] was the first paper to analytically study the performance of P2P systems. They modeled P2P systems with multiple class queuing networks, where peers with different behavior belonged to different classes (e.g. the class of general peers and the class of free-riders). In their model, there was a single queue serving all queries from different peers, while each sharing file was served by its own service queue. Thus, the rate about servicing the queries was in proportion to the number of online peers, while the capacity of servicing a sharing file was

proportional to both the number of online peers and the popularity of the file. With these assumptions, the service capacity of a distinctive file was described by the Zipf's distribution [48], and system throughputs under distinctive P2P architectures were compared: queries through centralized nodes, queries through flooding, and queries through DHT. However, their assumption of "a newly arriving peer being available to serve files to its peers" is not true in reality, and this can negatively affect the accuracy of this model.

M. Lin et al. [49] proposed a stochastic framework to investigate the performance of P2P file-sharing. Based on the density dependent jump Markov process model [50], they studied the performance of P2P file-sharing systems where a file with K chunks was distributed among a number of peers. In their model, the system states were described by peers having different number of chunks. Using the theory of Markov process, the authors deduced the formulas for a system's average downloading time. They also provided the upper bound and lower bound of the average downloading time and sojourn time of a peer when the peer's downloading file included a total of K chunks. However, the assumption of new arrival peers always holding one chunk limited the application scope of this model.

In addition to these studies in general P2P file-sharing systems, the performance of BitTorrent and aMule/eMule, the largest P2P file-sharing systems in the real world, have also been widely analyzed.

D. Qiu et al. [51] developed a fluid model to study the performance of BitTorrent. In their model, peers were divided into downloaders and seeders. The change rates of downloaders and seeders were formulated by the differences between their arrival rates and departure rates. The system average downloading time was derived by Little's law [52]. With their analytical results, the authors believed that BitTorrent was very efficient at distributing files.

L. Guo et al. [53] adopted the same model in Dongyu's work to study the evolution of a single-torrent system. They found the service availability became poor quickly when the arrival rates of peers were reduced exponentially. After that,

they developed a graph model to analyze the correlation of a multiple-torrent system, where peers exchanged more than one file simultaneously. Finally, the authors suggested that the collaboration among multiple-torrent system could be used to solve the issue of service availability in a single-torrent system.

Y. Tian et al. [54] presented an analytical model to explore the distribution of peers who download a distinctive file in BitTorrent. With the extension of Dongyu's model, they explored a file sharing process with the Markov chain, with which the transfer rates between two adjacent states were a function of both the number of peers and the effect of TFT incentive policy [17] among them. Their numerical results showed that the distribution of peers at the different levels of their downloading completeness presented a U-shaped curve. They believed this was because BitTorrent's TFT policy limited the rate of peers obtaining content when they had very small or very large portions of the sharing file.

S. Petrovic et al. [55] also presented a fluid model to study the performance of eMule. By applying Little's law and the conservation law that the total uploading capacity equals the total downloading capacity, the model could build the relationship between the average download time of a file and the popularity of this downloading file. However, this approach has been limited by the assumption of each peer connecting to all the other peers in the system.

4.2.1.2 Experimental Approaches

Experimental methods have also been widely employed to study the performance of P2P systems in the real world. These methods usually measure the real P2P systems for a relatively long period with either passive monitors or active crawls.

M. Izal et al. [56] analyzed a 5-month track log, which was recorded while a tracker monitored the content exchange process of peers. They observed that P2P file systems like BitTorrent had good scalability and high average downloading rates. Using a modified BitTorrent client to achieve a series of content exchanging processes, they also showed that a correlation existed between the downloading

rate and uploading rate.

J. A. Pouwelse et al. [57] collected the information of peers from a BitTorrent tracker and measured the downloading progress of these peers. Their measurements revealed that the average downloading rate of peers was high, the availability of sharing content depended on their popularity, and the availability of peers was low because of the deficiency of the incentive policy to seeders.

J. Yang et al. [58] adopted an active probing method to measure aMule/eMule. They developed several applications to crawl hundreds of index servers and millions of eMule/aMule clients separately. With these measurements, they presented the geographic distribution of these clients and the capacity of each index server.

L. Plissonneau et al. [59] focused on a large number of residential aMule/eMule users with ADSL connection. By monitoring the TCP connection of these customers, they distinguished the aMule/eMule's flows from others via their specialized transfer ports. With the measurements, they revealed that the popularity of sharing files were not Zipf distributed, while the waiting time before downloading was correlative with the popularity of the downloading file, and the downloading rate was not as fast as the connection capacity.

4.2.2 Research on Fairness

Due to their self-organizing and self-managing features, the system performance of P2P file-sharing networks fully relies on each peer's cooperation on a fair foundation. As a result, a number of studies have also focused on the fairness issue in P2P file-sharing networks.

4.2.2.1 General Studies on Reciprocity-Based Incentive Policies

To maintain fairness in a distributed P2P system, reciprocity incentive policies might become a fundamental component. The relative studies are summarized below.

D. S. Menasche et al. [4] focused on studying the efficiency of reciprocity in-

centive policy in a general P2P file-sharing system. By defining the reciprocity mechanism into two types: direct and indirect, they compared the efficiency difference between them. Their analytical results showed the efficiency of an indirect reciprocity mechanism could reach at most twice of the efficiency of a direct policy. However, their classing the incentive policy of aMule/eMule into the indirect reciprocity type is not believed to be correct.

M. Feldman et al. [5] studied the reciprocity mechanism of P2P file-sharing systems with game theory. The authors presented that an indirect reciprocity mechanism could provide a higher level of cooperation to a system than a direct reciprocity mechanism. They were also the first to discuss the stranger policies for P2P systems and pointed out that inappropriately dealing with strangers could collapse the system. To deal with the whitewashing issue, the authors also proposed a stranger adaptive policy, but this policy had only been verified by simulation.

B. Q. Zhao et al. [14] designed a general framework to analyze the incentive policy in P2P networks. In this framework, the incentive policy was modeled with the gain obtained by a peer when this peer contributed service to other peers. However, the prediction of their model about the mirror incentive policy (like BitTorrent's TFT policy) leading to system collapse had not been founded in the real world. Moreover, the impact of stranger policy had not been considered in this work.

4.2.2.2 Indirect Reciprocity Incentive Mechanism

As a reminder, an indirect incentive mechanism can help a peer choose their partners according to their individual global reputation levels. The following part summarizes the relative studies.

R. Sherwood et al. [11] designed a cooperative system for P2P with the indirect incentive mechanism. In this system, each transaction was assigned a trust value. When calculating the trust level of a single peer, a peer first built a trust graph including multiple direct or indirect transaction paths from itself to this peer.

After that, the trust value was computed with the combination of the transaction values along different paths. However, how to deal with newcomers had not been considered within this work.

S. D. Kamvar et al. [12] presented an algorithm to assign a global trust value to every peer. The local reputation score of each peer would first be computed by itself independently, and accordingly each peer's global reputation value was determined by using the local scores from its partners, weighted by the global reputation of these partners. A distributed algorithm was then developed, which allocated the computing and storing tasks to each peer. Unfortunately, the newcomer issue had not yet been discussed in this work.

Z. Y. Liu et al. [13] designed an indirect incentive algorithm based on the assumption that each peer in the system belonged to a social network. By using the knowledge of the underlying social network, the authors showed that this algorithm had a good efficiency under different service demands.

4.2.2.3 Direct Reciprocity Incentive Mechanism

The direct reciprocity incentive mechanism, due to its simplicity, has also been adopted by popular P2P file-sharing systems like BitTorrent and aMule/eMule. However, because of some design issues, both BitTorrent and aMule/eMule did not provide sufficient incentive to peers and therefore could not efficiently limit free-riding behavior. The related research is summarized below.

T. Locher et al. [60] developed a free-riding BitTorrent client called BitThief to examine the efficiency of the official Tit-For-Tat incentive policy. Using some design issues of BitTorrent, this client could retrieve many more sources from trackers than the official BitTorrent client. As a result, their experiments showed that a BitThief client could always achieve a high downloading rate, even though it never contributed any data to others.

M. Sirivianos et al. [18] also modified a BitTorrent client that knew more peers than the compared official client. Like Locher's work, they also showed

that this modified client achieved free-riding easily, and its average downloading rate was higher than that of the compared official client. Using PlanetLab-based experiments, they indicated that the system performance would be degraded with the increment of the population of the free-rider.

S. Handurukande et al. [20] modified an eDonkey client and used it to crawl the eDonkey network for over 50 days. Besides showing the existence of the clusters, their work also found a popular free-riding behavior in the system. However, the reason for the popularity of free-riding in eDonkey network had not been pointed out in this paper.

4.2.3 Research on the Relation Between Performance and Fairness

A few researches [8, 9, 61, 62] have investigated the relation between system performance and system fairness as shown below.

R. Krishnan et al.[8] considered the shared content as public goods. During the content exchanging process, each peer tried to maximize its own utility. Using game theory, the authors proved that free-riders could still exist under the socially optimal outcome. Their conclusion is consistent with our work's observation, where optimal system performance comes from allocating partial bandwidth to strangers instead of extremely punishing free-riders.

M. Feldman et al. [9] modeled the impact of incentive policies on system performance on the basis of game theory. The author showed that system performance would be improved by punishing free-riding. The authors also considered that the punishing free-rider policy could not avoid restricting newcomers, but this only caused the decrease of system performance when the turnover rates were high. However, the authors had not studied the system fairness explicitly and quantitatively. Moreover, no verification has yet been given in this work.

B. Fan et al. [61] did not focus on a general P2P network, but a specific P2P file-sharing system: BitTorrent. The trade-off of BitTorrent's performance and

fairness was further investigated via an uplink-sharing fluid model, where system performance was assumed to be decided by the total uploading rate. With both the analytical and simulation results, the authors considered the design of BitTorrent emphasized fairness more than performance.

4.2.4 Summary of Related Work

Even though a great amount of researches focusing on studying the performance and the fairness of P2P system already existed, the stranger issue, during their studies, had either been inappropriately ignored or not been well modeled. Inspired by the previous works, we investigated the stranger issue in general P2P file-sharing systems in the remainder of this chapter.

4.3 Stranger Policies Under the Indirect Reciprocity Mechanism

Indirect reciprocity mechanisms (IRMs) are recognized as being more scalable, but more complex than direct reciprocity mechanisms (DRMs). The global reputation system is the most popular application of IRMs. It has been popularly proposed to maintain the fairness of P2P systems in [27, 12, 11, 28]. In a P2P file-sharing network with a global reputation system, each peer will normally receive a unique reputation score according to their previous sharing experience. Since this information is globally available, each uploader will allocate their uploading bandwidth to other peers according to their scores. With the global score, general peers can be easily rewarded, and free-riders can easily be punished. If some peers pursue benefits by sharing content for a period and then free-riding, their counterfeit behavior could be easily distinguished and restricted by assigning different weights to the long-time behavior of peers and their short-time behavior in the calculation of their reputation score. However, the IRMs are usually vulnerable to whitewashing [3, 9, 46] because whitewashers cannot be distinguished from general newcomers.

Therefore, stranger policies under the IRMs may either compromise the penalty to whitewashers, or delay the cooperation of general newcomers. In this section, we explore the impact of different stranger policies with numerical analyses and agent-based simulations.

4.3.1 An Analytical Model

To study stranger policies, we conduct mathematical analyses based on the fluid modeling method. The fluid-based models were popularly adopted in [49, 51, 54, 55, 61] for studying P2P file-sharing systems. In this work, We modify the classic fluid model that assumes peers provide the same uploading bandwidth during the whole content exchange process. Specifically, we take into account the amount of available uploading bandwidth that each resourceable general peer (the uploader) can provide, when these peers experience different content exchange periods.

4.3.1.1 Model Description

In a P2P file-sharing system with the IRM, peers are mainly divided into two classes: strangers and known peers. The strangers include general newcomers and whitewashers, and the IRM has no information about these peers. It is worth noting that free-riders must whitewash for survival in a P2P system with the IRM. Otherwise, free-riders without whitewashing could be easily recognized and isolated. We assume that, to obtain the benefits intended to strangers, free-riders have strong motivations to whitewash, and consequently they are all whitewashers. The known peers are peers with sharing experiences. Since free-riders without whitewashing are isolated by the IRM, known peers are considered to be legitimate peers. In other words, they are resourceable general peers. With this classification, different stranger treatment policies can be evaluated according to the truth that each resourceable general peer independently assigns different percentages of uploading bandwidth to the known peers and the strangers respectively.

Our model considers a download task that requests each peer to download

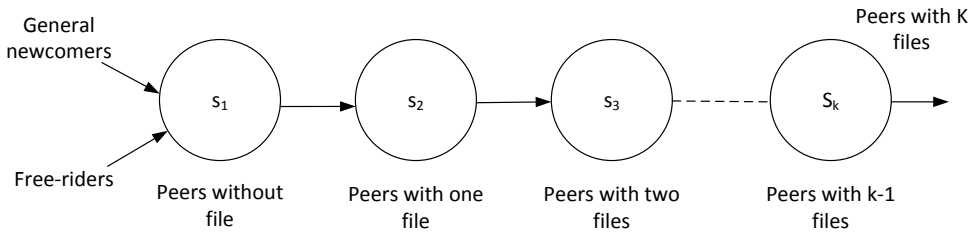


Figure 4.3.1: The analytical model assume peers need to download total K files. At state S_1, a peer will download its first file, then this peer moves to state S_2 and download the second file. The process continues until this peer downloads its last file at state S_k and leaves the system.

K files. As shown in Figure 4.3.1, a general peer arrives into the system as a general newcomer without any files downloaded yet. In state S_1 this general peer randomly chooses its first file for downloading from other peers. The general peer then moves to state S_2 where it has one downloaded file. As state S_2, the general peer randomly choose its second file for downloading, and it also uploads the first file to other peers simultaneously. Continuing these uploading and downloading operations, this general peer will eventually reach state S_K, where it downloads its last file and completes the entire download task. On the other hand, when a free-rider tries to download the same K files, it will only process the same downloading steps without uploading any content at each state S_i ($1 \leq i \leq K$). Meanwhile, the free-rider will change its ID for whitewashing at each state S_i when downloading a new file. Naturally, this method can also directly model a chunk-based P2P downloading process for one file with K chunks, where each chunk is continuously downloaded when a peer passes through each state S_i.

This model follows the assumption in [61]: peers dynamically join and leave the system. The average arrival rates of general peers and whitewashers are denoted by λ_g and λ_f respectively. Peers will stay in the system until completing the task of downloading all K files. As illustrated in Table 4.1, $g_i(t)$ and $f_i(t)$ represent

Table 4.1: Parameters for the analytical model

Parameters	Meaning
λ_g	average arrival rate of general peers
λ_f	average arrival rate of free-riders (whitewashers)
ε	percent of uploading capacity of a resourceable general peer allocated to the known peers
S_i	download state i, where i is an integer
K	the total number of files in the download job
$g_i(t)$	the number of general peers in state S_i $(i = 1, 2, ..., K)$
$f_i(t)$	the number of free-riders (whitewashers) in state S_i $(i = 1, 2, ..., K)$
μ	peer's average upload capacity
$p_{m,n}$	the probability that a general peer with m files can upload content to another peer with n files $(m, n = 0, 1, 2, ..., K-1)$
R_s	a peer's sharing ratio
R_c	a peer's comparing ratio
d_g	general peer's average downloading rate (performance metric)
u_g	general peer's average uploading rate
d_f	free-rider's (whitewasher's) average downloading rate
F_m	fairness metric

the number of general peers and whitewashers at state S_i respectively. Without loss of generality, each file is assumed to be the same size.

Some previous fluid-based models [51, 61, 55] assumed that the uploading capacity μ of each resourceable general peer is fully used. However, a general peer can not provide uploading service to other peers when its content is not useful to those peers. For example, if the content held by peer A is possessed by all other peers, the uploading capacity of peer A cannot be used, even though A prefers to provide service to others. Following the method in [49], we remove this capacity assumption and introduce a novel parameter $p_{m,n}$. Before explaining the definition of $p_{m,n}$, we make an assumption that all downloading files in the system are uniformly distributed among peers. We believe this assumption can be guaranteed by the policy used in our model: the downloading order of the total K files is random. The parameter $p_{m,n}$ indicates the probability that a general peer with m files can upload content to another peer with n files. Additionally, as peers in different states possess different amounts of shareable content, they have different abilities to earn a reputation score. For example, peers in state S_1

only possess one file, compared with peers in the following states, their ability of sharing this file with others and according earn scores is lower. Consequently, $p_{m,n}$ can also be used to represent different reputation scores of peers in different states.

To model different stranger policies, we introduce a parameter ε to represent the percent of uploading bandwidth of a resourceable general peer allocated to known peers. Consequently, $1 - \varepsilon$ represents the percentage of the uploading bandwidth allocated to strangers. Consequently, people can model different stranger policies by changing the value of ε. For instance, an extreme restricting stranger policy can be represented by choosing a large value of ε, while an extreme rewarding stranger policy can be represented by choosing a small value of ε.

Based on the principle of each peer's downloads coming from the other peers' uploads and according to the property of continuous Markov chains [50], the varying tendency of the number of peers in the state S_1 and S_i $(1 < i \leq K)$ can be described by the following differential equations. The definitions of the variables used in these equations are given in Table 4.1.

For whitewashers at state S_1 and state $S_i (i = 2, 3, ..., K)$, Equations 4.3.1 model the change in the number of whitewashers in each state. The first part on the right in Equation 4.3.1 represents the number of whitewashers flowing into the state, and the second part represents the number of whitewashers flowing out. It should be noted that the flow out from state S_i equals the flow into state S_{i+1}. To represent the whitewashers flow out of one state, their received uploading rates need to be summed. These rates are from the resourceable general peers at all the states except state S_1. These resourceable general peers allocate a fraction $1 - \varepsilon$ of their uploading rate to strangers. We use μ to represent the average uploading capacity of the general peers. The uploading rate of the uploader at state S_j up to the download peers at state S_i are modeled by $\mu p_{j,i}$. Since each resourceable general peer possibly uploads content to peers at any state, these rates for one given state are partitioned by the ratio of the number of whitewashers at this given

state to the total number of peers at all states.

$$\begin{cases} \frac{df_1(t)}{dt} = & \lambda_f - \frac{f_1(t)}{\sum_{j=1}^{K}(g_j(t)+f_j(t))} \sum_{j=2}^{K} g_j(t)(1-\varepsilon)\mu p_{j-1,0} \\ \\ \frac{df_i(t)}{dt} = & \frac{f_{i-1}(t)}{\sum_{j=1}^{K}(g_j(t)+f_j(t))} \sum_{j=2}^{K} g_j(t)(1-\varepsilon)\mu p_{j-1,i-2} \\ & - \frac{f_i(t)}{\sum_{j=1}^{K}(g_j(t)+f_j(t))} \sum_{j=2}^{K} g_j(t)(1-\varepsilon)\mu p_{j-1,i-1} \end{cases} \tag{4.3.1}$$

where

$$p_{m,n} = \begin{cases} 1 & if \quad 0 \le n < m \le K-1 \\ \\ 1 - \frac{C_n^{n-m}}{C_{K-1}^{m}} & if \quad 0 \le m \le n \le K-1 \qquad C_x^y \text{is the binomial coefficient} \end{cases} . \tag{4.3.2}$$

Similarly, we use Equations 4.3.3 to model the change in the number of general peers at each state S_1, and state $S_i (i = 2, 3, ..., K)$. The first part on the right in these equations represents the number of general peers flowing into the state, and the second part represents the number of general peers flowing out. The flow out from state S_i equals the flow into state S_{i+1}. The uploading rates to general peers at state S_i are from the resourceable general peers at all the states except state S_1. These resourceable peers allocate a fraction ε of their uploading rates to known peers. Meanwhile, general newcomers at states S_1 are recognized as strangers, and then these newcomers obtain downloading rates from the remaining $1-\varepsilon$ percentage. Likewise, the uploading rate of an uploader at state S_j up to download peers at S_i are modeled by $\mu p_{j,i}$. Furthermore, since each resourceable

general peer can possibly upload content to peers at any state, these rates for one given state are partitioned by the ratio of the number of general peers at this given state to the total number of peers at all states.

$$\begin{cases} \frac{dg_1(t)}{dt} = & \lambda_g - \frac{g_1(t)}{\sum_{i=1}^{K}(g_j(t)+f_j(t))} \sum_{j=2}^{K} g_j(t)(1-\varepsilon)\mu p_{j-1,0} \\ \frac{dg_i(t)}{dt} = & \frac{g_{i-1}(t)}{\sum_{j=1}^{K}(g_j(t)+f_j(t))} \sum_{j=2}^{K} g_j(t)\xi\mu p_{j-1,i-2} \\ & -\frac{g_i(t)}{\sum_{j=1}^{K}(g_j(t)+f_j(t))} \sum_{j=2}^{K} g_j(t)\varepsilon\mu p_{j-1,i-1} \end{cases} \tag{4.3.3}$$

where,

$$\xi = \begin{cases} 1-\varepsilon & if \quad i=2 \\ & \\ \varepsilon & otherwise \end{cases}.$$

4.3.1.2 Performance and Fairness Metrics

To measure system performance and fairness, we define measurement metrics as follows (These metrics are also used in the subsequent sections):

- Performance metric d_g denotes the average downloading rate of the general peers. A large value of d_g usually represents a high system performance.
- Fairness metric F_m combines two kinds of influencing factors: the sharing ratio R_s and the comparing ratio R_c. R_s indicates the difference of a general peer's cost from its payoff, and R_s represents the fairness from a single peer's viewpoint, while R_c reveals the system viewpoint on fairness by comparing the free-rider's average benefit with the general peer's average benefit. Since

the upper limit of R_s and R_c is one ($u_g \geq d_g$ due to the whitewashers' existence), the closer F_m is to one, the more fair the system is.

$$F_m = \alpha_1 R_s + \alpha_2 R_c = \alpha_1 \frac{d_g}{u_g} + \alpha_2 \frac{d_g}{d_g + d_f}, \tag{4.3.4}$$

where α_1 and α_2 are the weights of these ratios, and $\alpha_1 + \alpha_2 = 1$, $0 \leq \alpha_1 \leq 1$, $0 \leq \alpha_2 \leq 1$. According to different design policies, P2P designers may assign different weights to R_c and R_s. For example, they may regard the sharing ratio R_s as more practicable than the comparing ratio R_c, because peers themselves can directly measure their own downloading and uploading rate, and accordingly calculating R_s in the system can much easier be implemented than calculating R_c. Alternatively, some designers may argue that punishing free-riding behavior should be given higher priority than other design objectives, since the accumulation of free-riders will cause serious damage to the whole system. We believe an area of future studies could focus on the methodology of accurately assigning different weights via using social-science techniques.

Once the system reaches a steady state, the transfer rate into state S_i equals the transfer rate out of state S_i, and the left hand of Equation 4.3.3 and of Equation 4.3.1 will become zero. By incorporating this feature with another important property of P2P file sharing systems, i.e., total uploading rates of the system are always equivalent to its total downloading rates, we can calculate the parameters d_g, d_f and u_g respectively.

As shown in Equation 4.3.5, the general peer's average downloading rate, d_g, equals the sum of the uploading rates, which are coming from all resourceable general peers at states S_2 to S_K and allocated to general newcomers and the uploading rates allocated to known general peers, normalized by the total number of general peers.

$$d_g = \frac{1}{\sum_{j=1}^{K} g_j}\left(\frac{g_1}{\sum_{j=1}^{K}(g_j + f_j)} \sum_{j=2}^{K} g_j(1-\varepsilon)\mu p_{j-1,0} + \sum_{i=2}^{K} \frac{g_i}{\sum_{j=1}^{K}(g_j + f_j)} \sum_{j=2}^{K} g_j \varepsilon \mu p_{j-1,i-1}\right). \tag{4.3.5}$$

As shown in Equation 4.3.6, the whitewasher's average downloading rate d_f equals the total uploading rates, which are coming from all resourceable general peers at states S_2 to S_K, allocated to whitewashers, divided by the total number of whitewashers.

$$d_f = \frac{1}{\sum_{j=1}^{K} f_j} \sum_{i=1}^{K} \frac{f_i}{\sum_{j=1}^{K}(g_j + f_j)} \sum_{j=2}^{K} g_j(1-\varepsilon)\mu p_{j-1,i-1}. \tag{4.3.6}$$

As shown in Equation 4.3.7, the general peer's average uploading rate u_g equals the sum of the uploading rates allocated to general newcomers, the uploading rates allocated to known peers, and the uploading rate allocated to whitewashers, which are coming from all resourceable general peers at states S_2 to S_K, normalized by the total number of resourceable general peers.

$$\begin{aligned} u_g = \frac{1}{\sum_{j=2}^{K} g_j}\Bigg(&\frac{g_1}{\sum_{j=1}^{k}(g_j + f_j)} \sum_{j=2}^{K} g_j(1-\varepsilon)\mu p_{j-1,0} + \sum_{i=2}^{K} \frac{g_i}{\sum_{j=1}^{k}(g_j + f_j)} \sum_{j=2}^{K} g_j \varepsilon \mu p_{j-1,i-1} \\ &+ \sum_{i=1}^{K} \frac{f_i}{\sum_{j=1}^{K}(g_j + f_j)} \sum_{j=2}^{K} g_j(1-\varepsilon)\mu p_{j-1,i-1}\Bigg). \end{aligned} \tag{4.3.7}$$

It should be noted that the property of the total uploading rates of the system being equivalent to the system total downloading rate doesn't mean that the average downloading and uploading rates can keep the same equation of $ug = d_g + d_f$, because d_g, d_f and u_g are all average values, and different kinds of peers

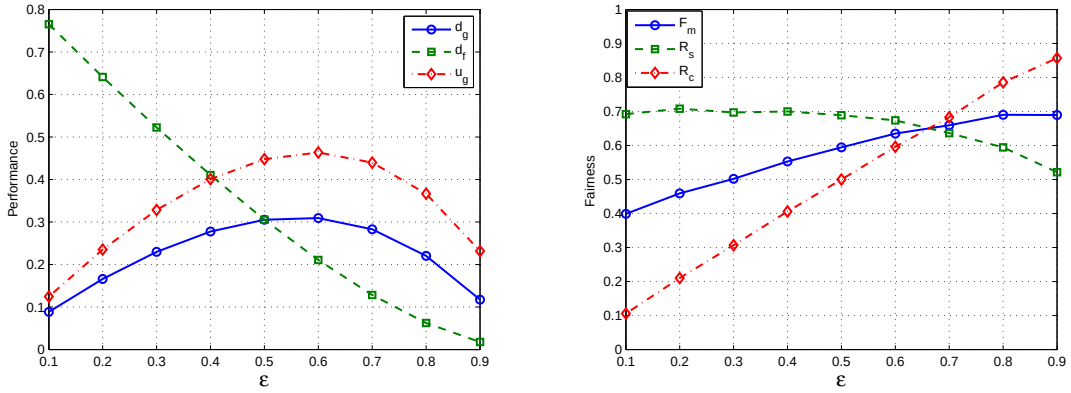
have a different population size.

4.3.2 Numerical Analysis and Verification

In this subsection, we use the mathematical model to numerically analyze the impact of different stranger policies on system performance and fairness. After that, we develop an agent-based simulation to verify the numerical results. The different stranger policies will be modeled by different values of the parameter ε.

4.3.2.1 Numerical Analysis

Using Equations 4.3.4, 4.3.5, 4.3.6, and 4.3.7, we numerically analyze the system performance and fairness of P2P file-sharing under different stranger policies. The related parameters in these equations are assigned as follows. We assign the parameter K the number 20, which means we focus on a file downloading process where each peer is required to download a total number of 20 files. Consequently, the total number of states in the model also equal 20. Without loss of generality, the size of each file is the same, and the average upload capacity μ of general peers is equal to one. During the time period of downloading one file, the arrivals of both general peers and free-riders follow Poisson processes with arrival rates $\lambda_g = 5/timeslot$ and $\lambda_f = 2/timeslot$. The values of ε are varied from 0.1, 0.2, 0.3, ... to 0.9. $\varepsilon = 0$ represents strangers getting all of the uploading bandwidth, and therefore, resourceable general peers are blocked. $\varepsilon = 1$ represents strangers getting none of the uploading bandwidth and accordingly general newcomers are blocked. Both $\varepsilon = 0$ and $\varepsilon = 1$ are impractical values for a P2P file-sharing system with an open environment, and consequently they are ignored in the following analysis. $\varepsilon = 0.1$ and $\varepsilon = 0.9$ represent the extremely rewarding stranger policy and extremely restricting stranger policy as shown in Figure 4.1.2 respectively, and the other values of ε represent the intermediate points in Figure 4.1.2, where strangers are assigned some uploading bandwidth. To calculate the fairness metric F_m, both α_1 and α_2 are assigned to 0.5, which means we give the same emphasis



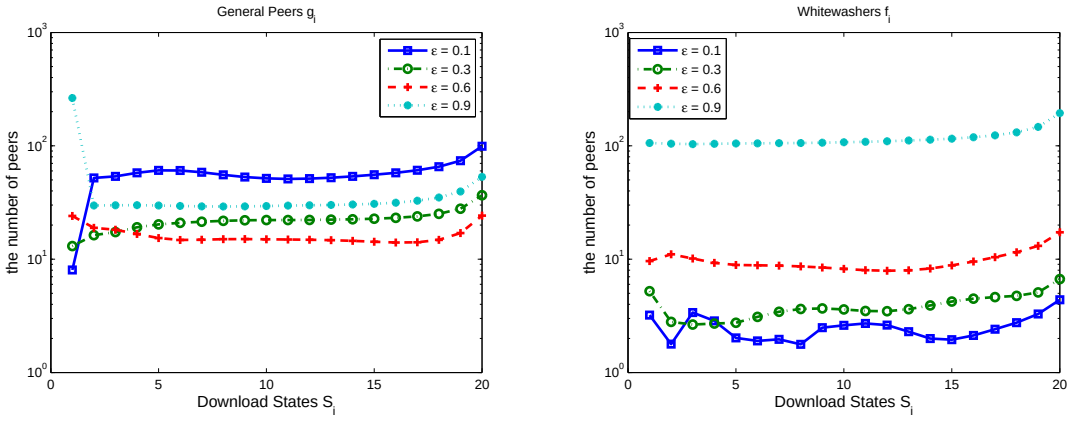
(a) (b)

Figure 4.3.2: Analytical results for different stranger policies: (a) shows the performance d_g and (b) shows fairness F_m as a function of ε (the percent of uploading bandwidth to known peers) when $\lambda_g = 5$ and $\lambda_f = 2$.

on fairness from both systems viewpoint and users' viewpoint.

Running our analytical model, we show changes of system performance in Figure 4.3.2(a). We discover that neither the extremely rewarding stranger policy nor the extremely restricting stranger policy can provide the highest performance. Instead, the largest value of the average downloading rate of general peers is obtained at one intermediate point of the value of ε being close to 0.6, and at that point 40% of uploading bandwidth is assigned to strangers. In addition, with less and less uploading bandwidth going to strangers (the increment of the value of ε), the average downloading rates of whitewashers keep falling. The results in Figure 4.3.2(a) also indicate that the contribution of general peers (their average upload rate u_g) is always greater than their benefits (their average download rate d_g). This makes sense because of the existence of both newcomers and whitewashers, who consume services without making any contribution.

The analytical results about fairness are shown in Figure 4.3.2(b). We find that the extremely restricting stranger policy helps the system achieve its best fairness, while the extremely rewarding stranger policy gives the worst fairness to the system. Obviously, restricting strangers also reduces the benefit of whitewashers, and promoting strangers improves the benefit to whitewashers at the same



(a) general peers (b) whitewashers

Figure 4.3.3: The number of general peers and whitewashers at different downloading states S_i ($i \in \{1, 2, ..., K\}$) under the different stranger policies (ε = 0.1, 0.3, 0.6, *and* 0.9 respectively) in the steady state.

time. Furthermore, the positive slope of the fairness metric F_m indicates that the system could distinctly limit whitewasher's benefit when the enforcement of restricting strangers increases. Regarding the changing trends of R_s and R_c shown in Figure 4.3.2(b), the increasing trend of R_c can be attributed to the continually decreased uploading bandwidth assigned to whitewashers with the growth of ε, while the decreasing trend of R_s can be associated with more and more general newcomers being delayed at the first state S_1.

Next, we clarify the impact of different stranger policies on the population of peers in the system. We verify the value of ε and plot the number of peers remaining at each state S_i in Figure 4.3.3 when the system reaches a stable state. If the number of peers remaining in the system is small, we can say that the peers leaving the system should have already obtained enough downloading bandwidth to complete their tasks and accordingly have left the system. If the number of peers remaining in the system is larger, peers may not able to get enough downloading bandwidth. In other words, the average downloading rate of peers varies inversely with the number of peers remaining in the system. Like Figure 4.3.3(a) shown, the total number of general peers remaining in the system when $\varepsilon = 0.6$ is smaller than those when we assign ε other values. Correspondingly, Figure 4.3.2(a) shows

that general peers obtain the largest downloading rate at this point. Similarly, the total number of whitewashers shown in Figure 4.3.3(b) monotonically increases with the increasing of ε, which is also consistent with the d_f's variation shown in Figure 4.3.2(a). Therefore, by showing the population of peers remaining in the system, we can provide convincing evidence to explain the results shown in Figure 4.3.2(a). In addition, when $\varepsilon = 0.1$, the number of remaining general peers is much larger than the number of remaining whitewashers. Under this extremely rewarding stranger policy, newcomers including whitewashers always receive far more bandwidth than known peers. When $\varepsilon = 0.9$, the remaining numbers of both general peers and whitewashers are very high. This is because the potential contributors are restricted at state S_1, and consequently they cannot contribute to others soon.

P2P designers usually have to decide which stranger policy may be suitable for their design goals: focusing on performance or paying more attention on fairness. We develop the following simple formula to help P2P designers calculate the design trade-offs DT.

$$DT = \alpha d_g + (1 - \alpha) F_m \tag{4.3.8}$$

To find an appropriate stranger policy, P2P designers can assign a weight α to system performance and accordingly assign $1 - \alpha$ to fairness respectively. For example, if designers pay most of their attention to system performance ($\alpha = 1$), according to Equation 4.3.8, they may assign an intermediate value to ε for their stranger policy as shown in Figure 4.3.4(a). By contrast, according to our analysis, restricting stranger policy at $\varepsilon = 0.9$ may be used when designers prefer fairness ($\alpha = 0$) as shown in Figure 4.3.4(b). Of course, as shown in Figure 4.3.4(c), designers can choose stranger policy $\varepsilon = 0.6$ for best trade-off, if they treat performance and fairness equally ($\alpha = 0.5$).

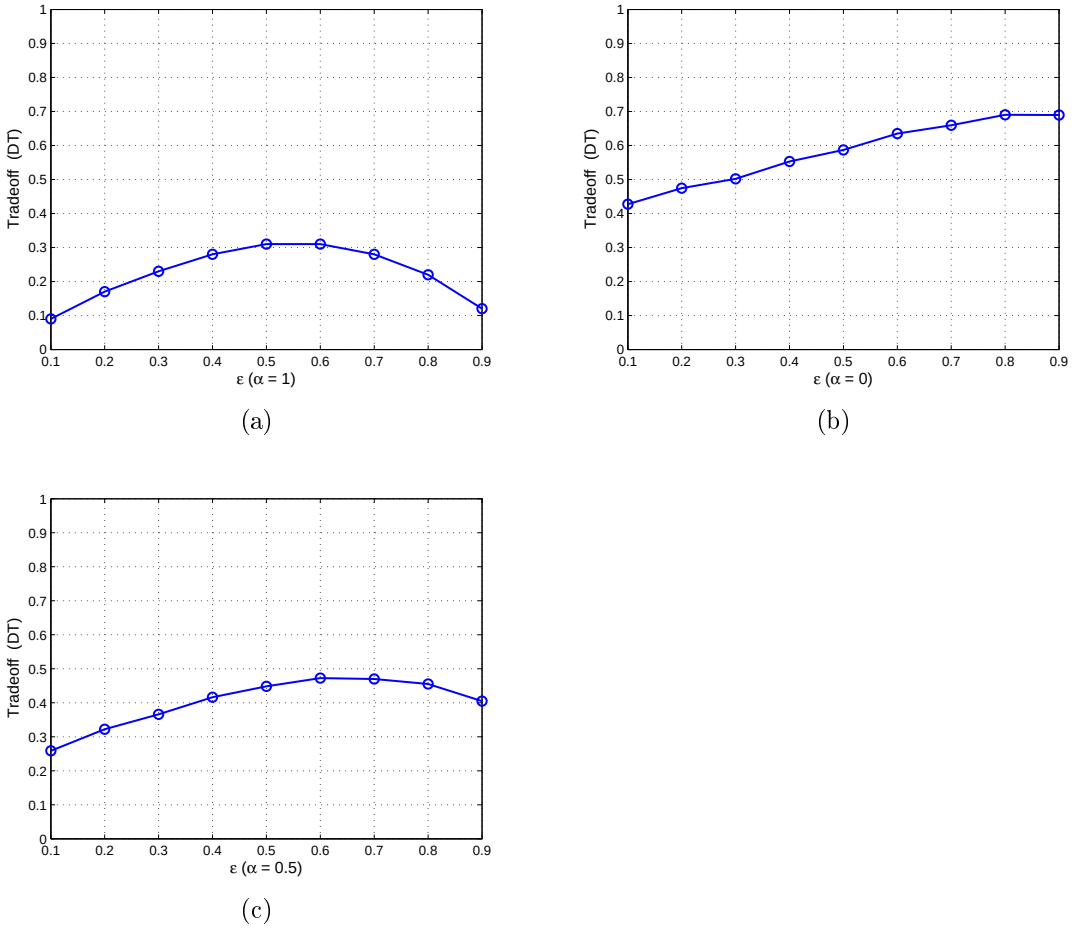


Figure 4.3.4: The trade-off of performance and fairness when designers choose different weight α.

4.3.2.2 Agent-Based Simulation

Using a typical agent-based simulation tool, NetLogo [63], we also develop an agent-based simulation to verify the analytical results. Agent-based simulation, as a class of computational modeling, has already been widely used in the research of ecology, economic, demography, etc. [64]. The agent-based simulation recently attracted the attention of P2P researchers [65, 66] for its capacity to simulate the behavior of autonomous agents and their interactions in a complex network.

Simulation Steps

In this simulation, we define two classes of agents coexisting in an open environment, general peers and whitewashers. The general peers can obtain reputation scores according to their contribution. The whitewashers will change their IDs

after downloading a file, so they have no reputation scores and are recognized as strangers. Each resourceable general peer maintains two uploading queues: one uploading queue holds the known peers, and the other holds strangers. Following the stranger policy, the resourceable general peer allocates its uploading bandwidth into two parts at any time; one part is allocated to a known peer chosen from the queue for known peers, and the other part is allocated to a stranger chosen from the queue for strangers.

To capture the functionality of the global reputation system, previous researchers have designed the IRM with either a decentralized or a centralized infrastructure [12, 11, 67, 68]. From the viewpoint of making the reputation score of each peer globally available to other peers, both centralized methods and distributed methods could provide the same ability. Peers in an IRM system have consistent opinions about who are known peers and who are strangers. However, both of these infrastructures also have their own issues in reality. For example, the centralized infrastructure may suffer from the scalability problem when there are millions of peers online, while the decentralized infrastructure faces the issue of how to synchronize the information effectively and quickly. Since our work does not focus on evaluating the structure of global reputation systems, we introduce a typical hybrid P2P infrastructure to simplify our simulation architecture. That is, a central index server is introduced into our simulation. It is globally available to every peer and responsible for maintaining the information of peers, such as the shareable content information and the reputation information of each peer.

We define the information publishing and retrieving processes as follows. After a general peer downloads content from another peer, this general peer will regard this contributor as its known peer and send its identification information to the central index server. At the same time, as a resourceable general peer (the potential uploader), this general peer will also publish its content information into this server. Both of these pieces of information have been recorded in the server's database. Similarly, a downloader will request the uploaders' information about

desired content from the index server. The server replies to the requests from downloaders with a list of content providers who are randomly selected from the server's database. After receiving the reply from the index server, downloaders can send content-downloading requests directly to these content providers. To deal with the downloading requests from different peers, each uploader will try to obtain reputation information about each downloader from the index server. If the reputation information of the requesting peer has been obtained from the index server, the uploader will insert this requesting peer into its uploading queue for know peers. Otherwise, it will add the requesting peer into its uploading queue for strangers, if the downloader is unknown. Additionally, in this simulation we assume the published information of each peer is trustworthy and ignore the possible attacks to the global reputation system such as peers lying about knowing other peers, Sybil attacks, or coalition attacks. There are many proposals about how to maintain the belief of each peer and how to deal with these attacks, but those are beyond this work's scope.

Table 4.2: Agent-based simulation parameters

Name	Value
files need to be downloaded	20
each file size	1000KB
general peers average arrival rate	5/time slot
free-riders average arrival rate	2/time slot
simulation time slot	100 sec.
peers maximum uploading capacity	20KB/sec.
initial peers in the system	100
1-ε: percentage of the uploading bandwidth of a resourceable general peer to the strangers	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9

We explore different stranger policies by running simulations with different values of ε ($\varepsilon = 0.1, 0.2, ..., 0.9$ respectively). Each uploader will choose two peers from its two queues respectively and upload to them according to the assigned stranger policy. The simulation parameters are shown in Table 4.2. The whole simulation time is divided into discrete time slots, and each time slot represents 100 seconds. In the simulation, new joined peers are required to download a total

of 20 files with the same size of 1000KB. Initially, there are 100 peers in the system. Each of them holds an arbitrary number of the downloading files, but the maximum number of held files is less than 20. By following a Poisson process, general peers and whitewashers arrive at the system with average arrival rates of 5/time slot and 2/time slot respectively. We ignore the possible downloading failure due to computer issues or transfer lines problems, and we, by following the previous work [61], also believe the probability of peers' voluntary departure when they are downloading their desired content is small. As a result, we assume each peer will stay in the system until they complete the entire downloading task.

At the beginning of a time slot, each resourceable general peer will randomly choose a known peer and a stranger from their uploading queues respectively. Then, the resourceable general peer separates its whole uploading bandwidth according to the stranger policy and uploads to both the known peer and the stranger. For the current mainstream access links, such as the ADSL and cable modem, the downloading bandwidth is no less than the uploading bandwidth. Thus, we assume the downloading rate is not the limiting factor for the simulation. Instead, we assume each peer has a 20KB/second uploading capacity for simplicity, and this is the maximum uploading rate that an uploader can provide. However, as we mentioned in the previous section, the uploading capacity may not be fully used, and the uploading rate of each resourceable general peer will be dynamically changed by the amount of their holding content. As a result, a downloader may have a very high downloading capacity, but this capacity may not be fully used during its downloading process. We believe this assumption is closer to the situation in the real world. For instance, considering the following situation: if downloader B has all the content that uploader A owns, A is not able to provide uploading service to its partner B. Therefore, A's uploading rate greatly depends on the total shareable content it possesses, and the uploading rate of a general peer will increase with the amount of its holding content. However, we also decide that the uploading rate is still limited by the defined uploading

capacity of 20KB/second.

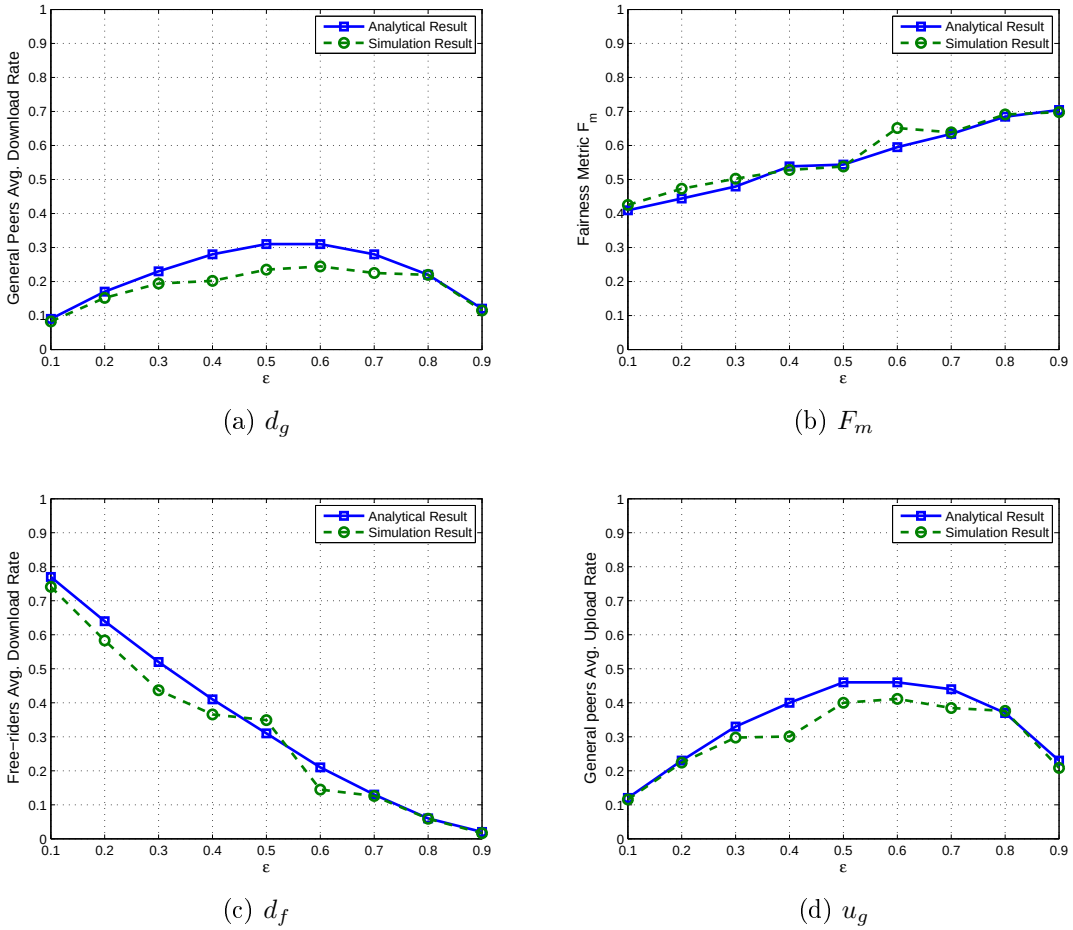
We ran this simulation on a standard desktop PC with Intel i7 core and 8GB memory. Each round of the simulation was about a total of 15000 cycles, which approximately equaled 17 days (one cycle represents 100 seconds). Since we logically simulated the uploading and downloading process within each time interval (there is no real downloading/uploading during the simulation), the actually running time for simulating each stranger policy was between 1 to 2 days. During each simulation, more than 100,000 peers attended the system for data exchange, and approximately 4000GB of data were simulated for downloading and uploading. During the simulation, we recorded each general peer's uploading rate and downloading rate and each whitewasher's downloading rate. At the end of the simulation, we calculated the average value of these uploading and downloading rates respectively.

Simulation Results

Both analytical results and simulation results about d_g, F_m, d_f, and u_g are compared in Figure 4.3.5(a), (b), (c) and (d) respectively. As shown in these figures, the simulation results are consistent with the analytical results in the previous section. Therefore, this verification confirms that our analytical model can capture the characteristics of stranger policies and can be used for the design of future P2P stranger policies. Most importantly, we have the following findings of both the numerical analysis and the simulations:

- The highest system performance comes from a trade-off of allocating uploading bandwidth to both known peers and strangers, but not from an extremely promoting stranger policy.
- The best fairness is reached when applying an extremely restricting stranger policy, while system performance is poor at this point.
- The variations of performance and fairness are different.

There are some negligible differences between the analytical and simulation results.



(a) d_g (b) F_m

(c) d_f (d) u_g

Figure 4.3.5: Results of a comparison of analytical modeling and agent-based simulation under different stranger policies when $\lambda_g = 5$ and $\lambda_f = 2$.

We attribute these differences mainly to the downloading pattern of peers used in the simulation. In fact, in the simulation, we assume that if a peer completes downloading a file before the end of a time slot, the peer will be idle until the next time slot begins.

4.3.3 Stranger Policies Under Different Whitewashers Population Size

In this section, we attempt to provide more insights on the trade-off between performance and fairness by using the mathematical model in Equations 4.3.4 4.3.5, 4.3.6 and 4.3.7. Our previous analysis revealed the effect of stranger policies on system performance and fairness, when general peers were the majority of the

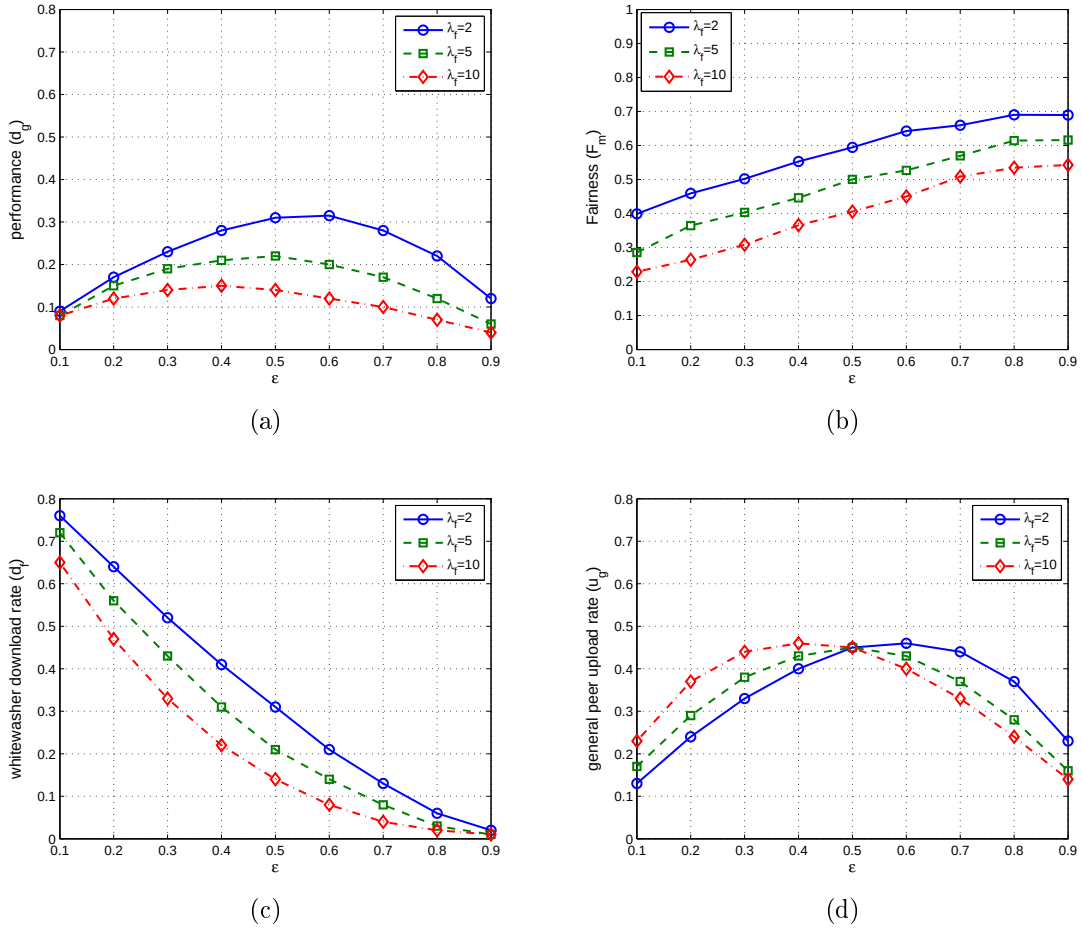


Figure 4.3.6: Analytical results for different stranger policies: (a) and (b) shows the performance d_g and fairness F_m as a function of ε , (c) and (d) show whitewashers average downloading rate and general peers average uploading rate respectively.

whole system ($\lambda_g \gg \lambda_f$). Naturally, we also want to know what will happen for system performance and fairness when whitewashers begin to dominate the system, because this scenario is crucial and has been discovered in current P2P file-sharing systems [2]. To answer this question, we run the mathematical model under the following two scenarios where the general peer's arrival rate is still $\lambda_g = 5/timeslot$, but the whitewasher's arrival rates become $\lambda_f = 5/timeslot$ and $\lambda_f = 10/timeslot$ respectively.

The analytical results are shown in Figure 4.3.6. The curve of both performance and fairness as functions of ε keeps the same shape of the previous analytical results, when the arrival rate of whitewashers was low. However, by comparing

the exact values of d_g and F_m under the same stranger policy, we find overall system performance and fairness decreases with the increase of whitewashers as shown in Figure 4.3.6. We attribute this to the growing trend of whitewashers and to the fact that whitewashers cannot be distinguished from general newcomers under IRMs. Furthermore, we find the value of ε for reaching the highest performance become smaller with the increment of λ_f, and consequently at these optimal points the strangers will receive more uploading bandwidth. When the arrival rates of whitewashers increase, keeping the same value of ε will cause less bandwidth assigned to general newcomers, who are competing with more and more whitewhashers. Accordingly, this will reduce the average download rate of general peers. On the contrary, decreasing the value of ε will allocate more bandwidth to strangers. Consequently, this will support general newcomers obtain uploading bandwidth relatively quickly, and it will bring more new contributors to the system early. Unfortunately, as Figure 4.3.6(c) shown, rewarding strangers also benefits whitewashers, who may gain identical or even higher average download rates than general peers at those optimal performance points.

4.3.4 Summary

Both our analytical and simulation results reveal that, with stranger policies under the IRMs, the highest system performance and the best fairness cannot be reached at the same time. The highest performance can be obtained via a trade-off between rewarding strangers and restricting them, and the trade-off level depends on the population size of free-riders. In other words, free-riders will survive when the highest performance is reached, which is consistent with the previous discovery in [8], where the authors mathematically proved that free-riders can still exist under the socially optimal outcome. On the other hand, the extremely restricting stranger policy could bring the best system fairness, but it also produces the lowest performance when the load of free-riding is high. In contrast, the extremely rewarding stranger policy cannot provide the highest performance to the system,

and it leads to the lowest performance when the load of free-riding is low, additionally, it also degrades the system to the worst fairness. Therefore, we suggest P2P designers, when designing the indirect reciprocity incentive mechanisms, carefully choose their stranger policies according to their individual goals.

4.4 Stranger Policies Under the Direct Reciprocity Mechanism

In this section, we study the stranger policies under the DRMs. The direct reciprocity incentive mechanisms are used in current popular P2P file-sharing systems such as BitTorrent and aMule/eMule. In these P2P file-sharing systems, each peer independently exchanges resources with other peers on the basis of its own individual experience. Compared with the IRMs, the DRMs simplify system implementation and reduce network traffic. However, a resourceable general peer under the DRM may be recognized as a stranger by another resourceable general peer if these two peers have not exchanged content before. This is because resourceable general peers will differentiate their partners only in accordance with their local historical knowledge. On the other hand, unlike a P2P system with the IRM, a free-rider in a P2P network with the DRM can easily obtain the benefits intended for strangers without whitewashing. The reason is that the local knowledge of general peers is not spread to others, and the current P2P file-sharing systems have no mechanism to help peers find their previous partners. As a result, free-riders do not need to worry about their selfish behavior toward one peer being known and consequently being punished by other peers. Thus, whitewashing also becomes unnecessary in a P2P file-sharing system with a DRM. This easy free-riding behavior has been confirmed by the experimental studies in [2, 18, 21].

Instead of mathematically modeling stranger policies in P2P file-sharing systems with the DRMs, we explore them directly through an agent-based simulation model for accuracy. Otherwise, the probabilities of resourceable general peers be-

ing recognized as strangers need to be adopted, but this probability is believed to be very dynamic and will be changed according to many different factors. Following those simulation results, we also conduct two case studies about the stranger policies in both BitTorrent and aMule/eMule.

4.4.1 An Agent-Based Simulation Model

As defined in the previous section, each resourceable general peer in this agent-based simulation model classifies its partners into two parts - known peers and strangers. We believe this classification simplifies the evaluation of stranger policies. The known peers for a resourceable general peer are its previous partners uploading content to it. But the resourceable general peer keeps the information of its known peers locally instead of spreading them globally through a central server. On the other hand, from the viewpoint of the resourceable general peer, its stranger may be categorized into 3 types:

- a general newcomer,
- a resourceable general peer without exchanging content before,
- or a free-rider.

Accordingly, each resourceable general peer independently decides the percentage of its uploading bandwidth to strangers based on the assigned stranger policy.

During the simulation, the central index server is retained. However, unlike our previous agent-based simulation on stranger policies under the IRMs, this index server is only in charge of the task related to publishing and retrieving content information. There are other content searching methods like decentralized-based query-flooding and DHT, but searching content centralized or distributed does not affect the usage of stranger policy. Moreover, searching content from central nodes is still the main method in BitTorrent, and the current design of DHT prevents them from effectively being used [69]. Thus, we ignore the other content searching methods in the simulation implementation and only adopt the central

index server for searching content. In this simulation model, the central server is not responsible for collecting the information of known peers. Instead, each peer individually maintains its known peers through its local database. Specifically, when a general peer A downloads a file from another peer B, it will assign a reputation score to B (this score will be increased when A downloads more content from B) and record B as a known peer in its local database. If B requests content from A later, A's uploading bandwidth to known peers will be allocated back to B according to its own data records rather than consulting with the central index server. In addition, a free-rider in this model is not a whitewasher any more, so the free-rider will never change its ID during the whole downloading task.

The remaining settings in the simulation model are the same as in our previous agent-based simulation under IRMs. Each resourceable general peer maintains two uploading queues: one for known peers and the other for strangers. In each time slot, this resourceable general peer assigns ε percentage of its uploading bandwidth to the peer with the highest reputation score chosen from its uploading queue for known peers, while it assigns the remaining uploading bandwidth to a peer which is randomly selected from its uploading queue for strangers. The downloading rate of each peer is not the limiting factor. The uploading rate of each resourceable general peer varies with the total amount of its holding content, but the uploading rate is limited by the uploading capacity 20KB/second. The simulation parameters are shown in Table 4.2.

We also ran the simulations for different stranger policies ($\varepsilon = 0.1, 0.2, ..., 0.9$ respectively). The general peer's arrival rate was still $\lambda_g = 5/100sec$, but the whitewasher's arrival rates changed based on the following values: $\lambda_f = 2/100sec$, $\lambda_f = 5/100sec$ and $\lambda_f = 10/100sec$ respectively. Each simulation ran for a total of 15000 cycles, which approximately equaled 17 days (one cycle represents 100 seconds). For each simulation, more than 100,000 peers had participated in the system for content exchange, and approximately 4000 GB of content had been simulated for downloading and uploading. During the simulation, we recorded

each general peer's uploading rate and downloading rate and each free-rider's downloading rate. At the end of the simulation, we calculated the average value of these uploading and downloading rates respectively.

4.4.2 Simulation Results

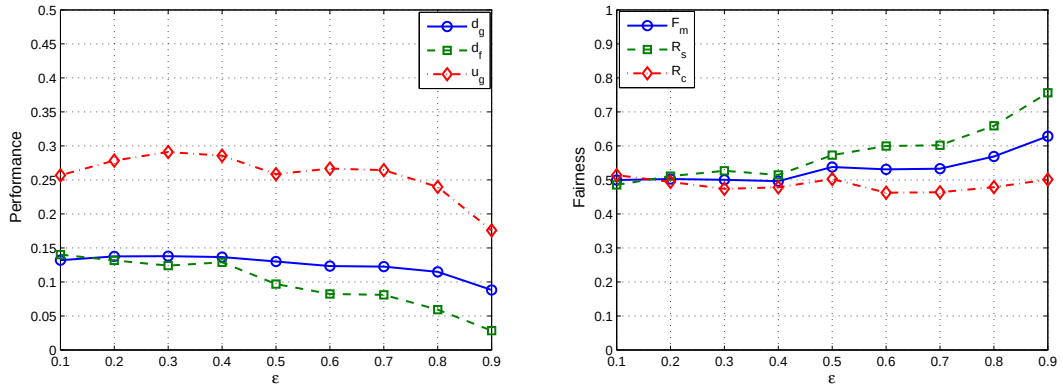
The simulation results under different loads of free-riders are shown in Figure 4.4.1. We first illustrate some intriguing findings which are different from the previous discovery under the IRMs.

- The curve of system performance d_g stays relatively flat until a high level of restricting strangers policy causes d_g to decline sharply. Moreover, free-riders cannot obtain a much higher downloading rate than general peers in the condition of more preferring to reward strangers ($\varepsilon < 0.5$). The reason is that in contrast with all resourceable general peers always being recognized as known peers under the IRMs, some of them are recognized as strangers under the DRMs. Accordingly, general peers can always be allocated bandwidth whenever rewarding stranger policies or restricting stranger policies are adopted. So the values of $d'_g s$ are relatively similar. On the other hand, d_g has a sharp decline because extremely restricting strangers will delay the potential cooperation of general newcomers and consequently reduce the whole number of uploaders. As a result, the average downloading rate of general peers is negatively affected. To explain the reduced downloading rate of free-riders, we believe that some resourceable general peers being recognized as strangers will compete for downloading bandwidth with free-riders, and consequently, this competition reduces the average downloading rate of free-riders. It should be pointed out that the above analysis cannot be evidence of the DRMs being superior to the IRMs. Since the DRMs reduce the cost of free-riding when compared with IRMs, they may attract more free-riders to the system.

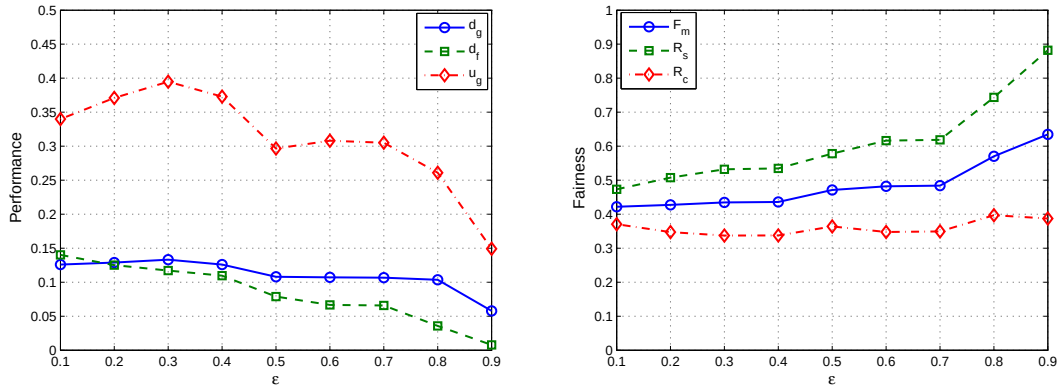
- The overall system performance declines little compared with the significant increase in the load of free-riders. When general peers compete with more and more free-riders, they obtain less downloading bandwidth and consequently must stay longer in the system to complete their tasks. This conversely extends the contribution of general peers to the whole system (This can be verified with the growing trend of general peer's average uploading rate u_g shown from Figure 4.4.1(a) to 4.4.1(b) to 4.4.1(c).). As a result, peers can benefit from general peers' staying longer.

- System fairness also stays relatively flat except when very high levels of restricting strangers is adopted. Even though rewarding stranger policies may provide benefit to free-riders, they are also able to benefit these resourceable general peers who are considered strangers. As a result, the concrete values of F_m under the DRMs in Figure 4.4.1 are higher than those under the IRMs in Figure 4.3.6 when rewarding stranger policies are assigned.

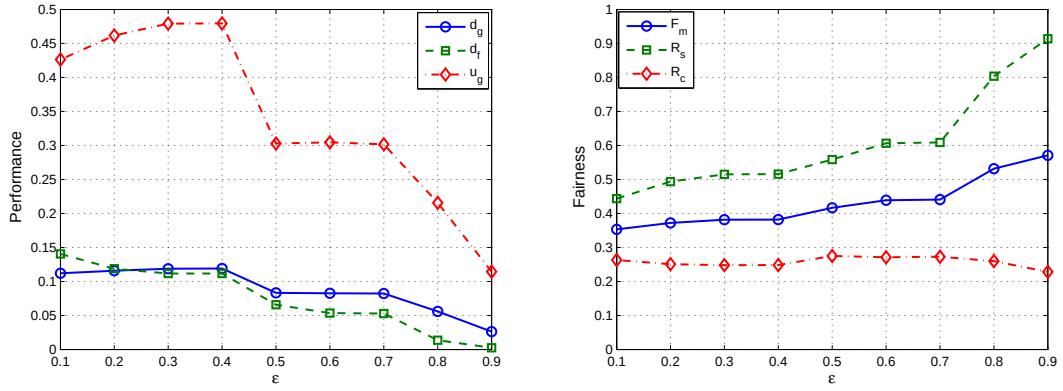
Certainly, these simulation results also show similarity in some respects to the previous observations under the IRMs. For example, the highest performance and best fairness of the system under the DRMs cannot be simultaneously reached. Specifically, the extremely restricting stranger policy also brings the best level of fairness at a cost of performance degradation. The extremely rewarding stranger policy also leads to the worst fairness. With an increase of free-riding load, both the system performance and fairness decrease. Thus, a delicate trade-off in dealing with strangers is also required. In summary, these similar characteristics of stranger policies under both the IRMs and DRMs will simplify the P2P incentive design in the future.



(a) $\lambda_f = 2$



(b) $\lambda_f = 5$



(c) $\lambda_f = 10$

Figure 4.4.1: Simulation results for different stranger policies under the DRMs: (a) shows the performance d_g and fairness F_m as a function of ε (the percent of uploading bandwidth to known peers) when $\lambda_g = 5$ and $\lambda_f = 2$; (b) is in the condition of $\lambda_g = 5$ and $\lambda_f = 5$; (c) is in the condition of $\lambda_g = 5$ and $\lambda_f = 10$.

4.4.3 Case Studies: Stranger Policies in Real P2P File-Sharing Systems

In this section, we elaborate the accuracy of this simulation model through analyzing the currently most popular P2P file-sharing systems: BitTorrent and aMule/eMule, which achieve different design goals by adopting different stranger policies.

4.4.3.1 Case Study 1: BitTorrent's Stranger Policy

BitTorrent adopts an incentive mechanism based on TFT strategies [17]. This incentive policy aims to maintain fairness in exchanging a single file. With the TFT incentive policy, each peer independently chooses its content exchange partners according to their current sharing behaviors. The information of generous or selfish behavior of partners is neither spread nor saved, so BitTorrent's incentive policy belongs to the DRM. To complete a content exchange process, a resourceable general peer divides its whole uploading bandwidth into several equal parts. Following the TFT policy, this resourceable general peer is going to find a fixed number of downloading peers (in the official BitTorrent specification, this number is set to 4 by default) and upload to them every 10 seconds. These downloaders, chosen with the TFT policy, are the peers from whom the resourceable general peer will obtain data with the highest download rates. In addition, the resourceable general peer also runs an optimistic unchoking strategy [17] every 30 seconds. With the optimistic unchoking strategy, one of the current downloaders will be replaced with a randomly selected requester, even though this requester may not have exchanged content with the resourceable general peer earlier. The goal of this unchoking strategy is to discover better potential partners and to promote newcomers.

Through the above analysis, we can deduct that known peers and strangers are treated differently in BitTorrent. The TFT policy is obviously responsible for how to provide uploading bandwidth to known peers; while the optimistic unchoking

strategy plays a role of how to allocate partial uploading bandwidth to strangers. We use "partial bandwidth to strangers", not the "full bandwidth", because known peers can also be selected by optimistic unchoking strategy and therefore possess some probability of receiving uploading bandwidth.

To explore the design goal of BitTorrent's fairness policy, we develop the following simple analysis. We assume the total number of known peers, general newcomers, and free-riders in the system are n_k, n_n, and n_f respectively. We also assume the total upload slots of a resourceable peer is m. The percent of uploading bandwidth to known peers ε can be directly estimated as follows:

$$\begin{aligned} \varepsilon &= \frac{(m-1)+(\frac{2}{3}+\frac{1}{3}\frac{n_k-m}{(n_k-m)+n_n+n_f})}{m} \\ &\geq \frac{m-\frac{1}{3}}{m} \\ &= 0.92 \qquad\qquad when \qquad m=4. \end{aligned} \tag{4.4.1}$$

Our estimated value ε is consistent with the result of the previous work in [70], where the authors indicated ε is equal to 0.94 by carefully modeling the BitTorrent protocol. After estimating the value of ε, we can analyze the fairness design goal in BitTorrent according to our agent-based simulation model. Based on our simulation results that large values of ε will lead to high fairness, we can conclude that fairness is preferred to performance in BitTorrent. This conclusion is identical to the discovery shown in [61], where the authors have mathematically modeled the design trade-offs of performance and fairness in BitTorrent. Their analysis and subsequent experiments also revealed that the optimal performance and fairness in BitTorrent cannot be reached simultaneously, and fairness in BitTorrent is more emphasized than performance.

Our simulation results also indicate that the large values of ε may bring poor system performance, which means BitTorrent performance can be further improved. Thus, we believe that BitTorrent performance could still be improved

when strangers are assigned a little more bandwidth. This can be easily implemented by allocating a few more downloading slots to the optimistic unchoking strategy, which could consequently reduce the value of ε. Meanwhile, we admit that BitTorrent's fairness cannot remain at the same level any more as shown in Figure 4.4.1.

Even though a restricting strangers policy is employed by BitTorrent, previous works [60, 18] showed that in the real BitTorrent network, free-riders may still obtain more benefit ($d_f \geq d_g$) than general peers. We believe this situation does not weak the usage of our simulation model. In fact, the benefit of free-riders shown in these researches comes from other design vulnerabilities of BitTorrent. For example, there is no limitation regarding the searching for sources frequency for a single peer in BitTorrent. As a result, free-riders can gather many more sources than general peers by looking up for sources more often, and accordingly, they can gain more downloading bandwidth than compliant peers [18]. These vulnerabilities degrade system fairness even if an extremely restricting stranger policy is adopted. Therefore, in order to implement a specific design goal such as keeping the system fairness in BitTorrent, other appropriate strategies are essentially to be used in conjunction with the proper stranger policies.

4.4.3.2 Case Study 2: aMule/eMule's Stranger Policy

As another very popular P2P file-sharing system, aMule/eMule attempts to maintain its fairness with a credit-based incentive system. Unlike the TFT policy in BitTorrent, credit in aMule/eMule is used to maintain the fairness of content exchange over a long-term period. The credit system regulates that the unconsumed credit can reside with peers for several months [8], so peers can use the credit when they download multiple files in the future. Since credit operations only exist between the service provider and its direct service recipients, the fairness policy in aMule/eMule belongs to DRMs. Considering the few investigations about the fairness strategy of aMule/eMule to strangers, in the following we study the stranger

policy of aMule/eMule with real world experiments and agent-based simulations.

Design Goal of Fairness Policy to Strangers

The credit system is designed to reward sharing behavior explicitly. Specifically, when downloader A finds uploader B with desired content, A sends downloading request to B. B, upon receiving the request, will put A into its uploading queue. A's position in the queue depends on its credit score C_s [16] obtained from B previously. The credit system uses the following equation to calculate the credit score:

$$C_s = R_m \times T_c \tag{4.4.2}$$

In this equation, T_c represents the duration that peer A stays in B's uploading queue, and R_m denotes the level of the previous contribution from A to B. More than that, R_m is normalized between 1 and 10, which not only avoids explicitly punishing any peer, but also tries to promote cooperation of strangers by assigning $R_m = 10$ to strangers. In other words, with the same uploading queue staying time, a stranger will normally obtain higher priority for downloading than a peer contributing to the uploader as before, but its contribution level is less than 10. Therefore, we can say that an extremely rewarding stranger policy is employed in aMule/eMule. According to our simulation results shown in Figure 4.4.1, we believe that the fairness in aMule/eMule is low, even though its performance under the current stranger policy can be high. In addition, we have the following predictions based on our simulation:

- Free-riders will have similar average downloading rates as general peers in aMule/eMule.
- The uploading rate of general peers is much higher than its downloading rate.

- Improving system fairness gradually will avoid a large reduction of system performance.

Experiment in Real aMule/eMule

To verify our prediction, we develop an experimental study in the real aMule/eMule.

- To implement a free-rider client, we modified one of the popular client versions of aMule/eMule, eMule $v0.49C$. This modified client will only download content from others and will not upload content at all. We also use an unmodified client eMule $v0.49C$ to represent the general peer. These two clients are set up to have the same download capacity and upload capacity. They are separately installed on two computers with the same configuration in the same local network.

- After choosing the same desired content, we enabled these two peers to join the real network at the same time and leave the system when their downloading jobs were completed. Using the sources searching algorithm in aMule/eMule, both peers randomly selected their partners with different uploading/downloading capacities and different distances during the tests. At the end of the downloading process, each peer's average downloading rate and average uploading rate were calculated.

As the credit system tries to maintain fairness among single file/multiple files downloading processes over a long time period, we accordingly implement seven different tests. Tests 1-5 represent measuring the downloading process of a single file with different sizes; Tests 6 and 7 represent measuring the downloading process of multiple files within a long-term period. In addition, in our tests the general peer and the free-rider are assigned non-credit at the beginning of tests, so we can evaluate our prediction with the same credit baseline for both of these two peers.

For the first 5 tests, the total amount of downloading content is within 100MB, 100-300MB, 300-600MB, 600-900MB, and 900-1800MB respectively. To keep the test accurate, we test each of them multiple times and sample a wide range of

downloading material of different types of popular non-copyright content (free Audio, free Video, free Books, etc.) for each trial. After each download, the content is moved out of the sharing folder, and the test peers' IDs are randomly recreated to remove the previous credit effect.

For the sixth test, we select a very popular Asian TV show series (15 episodes) which are new and published online on and off for about two months as the downloading resources. The total amount of downloading content is 4250MB. For the seventh test, the downloaded content is selected more generally from a large scope of popular non-copyright materials with different sizes. The total amount of content is 6040MB, and the total test time is about one month. During both of these two tests, each peer's ID is retained until they downloaded the last piece of content. Since the free-rider client does not change its ID during these two tests, these tests also attempts to verify our prediction: under the DRM free-riders do not need to whitewash.

Experimental Results

The experiments in the real aMule/eMule took place over more than two months from August 2009 to October 2009. Each peer downloaded over 100GB content. The experimental results are shown in Table 4.3, Table 4.4, Table 4.5, and Figure 4.4.2 respectively. As predicted with our simulation model, we find that under the extremely rewarding stranger policy:

- the free-rider receives a very similar average downloading rate as the general peer,
- and the average uploading rate of the general peer is much higher than its average downloading rate.

The accuracy of the simulation model is verified again by the match of the experimental results and our predictions. On the other hand, a concern about the efficiency of the fairness policy in aMule/eMule may be issued due to the ease of free-riding. Our simulation results indicate to set the value of ε relatively larger

Table 4.3: Test results in aMule/eMule

Test No.	1	2	3	4	5	6	7
Number of Chunks	≤10	10~32	32~65	65~97	97~194	461~563	71~699
Total Amount (MB)	0 ~ 100	100 ~ 300	300 ~ 600	600 ~ 900	900~1800	4250	6040
Test Trials	32	36	31	32	36	1	1
General Peer's Avg. Source No.	110	177	245	556	167	232	223
Free-Rider's Avg. Source No.	101	186	249	506	152	228	210
General Peer's Avg. Download Rate (KB/s)	17.84	30.23	32.44	65.96	42.08	47.5	35.18
General Peer's Avg. Upload Rate (KB/s)	35.4	85.2	144.4	118.4	83.0	85.6	60.43
Free-Rider's Avg. Download Rate (KB/s)	17.14	31.46	29.19	52.67	36.39	49	37.63
Free-Rider's Avg. Upload Rate (KB/s)	0	0	0	0	0	0	0

can mitigate this problem. That is, increasing the value of ε can restrict free-riding as well as to maintain the same level of system performance. This mitigation can be easily implemented in aMule/eMule, if each resourceable peer adaptively adjusts the value of the modifier R_m assigned to its strangers, instead of directly giving the maximum value 10 to them.

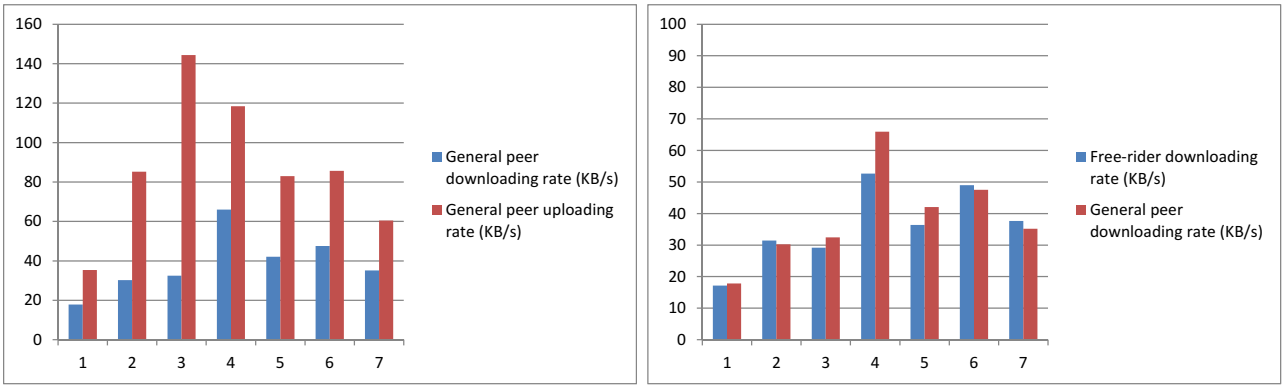
Fairness vs. Performance for aMule/eMule

With the prediction of our simulation results, we can avoid largely decreasing system performance while gradually improving system fairness. To control free-riding behavior, we could require each resourceable peer to adaptively adjust the value of the modifier R_m assigned to its strangers, instead of directly assigning the maximum value 10. That is, giving more uploading bandwidth to known peers.

By simulating aMule/eMule through the agent-based technology, we verify the impact of changing the stranger policies on the system performance. The simula-

Table 4.4: Detailed results of test 6 in aMule/eMule

Trial No.	File Size (MB)	General Peer's Download Rate (KB/s)	General Peer's Upload Rate (KB/s)	Free-Rider's Download Rate (KB/s)
1	295	31.59	81.12	26.68
2	328	42.18	84.46	44.31
3	348	67.31	87.26	79.09
4	311	40.88	88.99	42.25
5	293	64.72	89.09	55.98
6	315	45.65	84.62	55.38
7	286	51.02	86.22	59.6
8	305	48.19	87.98	48.85
9	289	65.65	81.84	66.4
10	299	35.67	89.26	35.16
11	296	33.74	74.33	25.47
12	296	55.25	86.46	58.94
13	288	44.38	85.42	49.91
14	286	42.54	85.93	42.98
15	289	43.75	87.81	44.56



(a) (b)

Figure 4.4.2: (a) general peer's average downloading rate vs. its average uploading rate; (b) general peer's average downloading rate vs. free-rider's average downloading rate

tor comprises two components: one ED2K index server node, and client nodes of aMule/eMule. The server node undertakes the publication of sharing content information and maintains the list of on-line nodes. Each client node independently exchanges content with others by running most of the functions of the client software of aMule/eMule. For example, each node in the simulator maintains its uploads by queuing technology. Each node will also perform aMule/eMule's inherent source searching methods such as eD2K server mehtod, Source-Exchange

Table 4.5: Detailed results of test 7 in aMule/eMule

Trial No.	File Size (MB)	General Peer's Download Rate (KB/s)	General Peer's Upload Rate (KB/s)	Free-Rider's Download Rate (KB/s)
1	388	62.12	102.19	92.88
2	427	19.19	30.18	41.78
3	433	14.37	30.23	12.38
4	394	41.29	66.36	29.7
5	328	56.26	78.01	55.22
6	377	5.46	10.00	4.95
7	365	31.86	55.49	23.91
8	348	38.59	70.66	48.15
9	385	25.84	52.37	28.59
10	419	61.01	110.08	60.02
11	364	38.63	50.95	34.21
12	446	28.93	55.77	25.6
13	407	7.64	12.27	6.5
14	347	16.25	35.05	58.78
15	135	17	30.22	15.24
16	252	55.15	55.15	57.72
17	699	60.41	97.85	63.73
18	74	31.75	70.24	23.24
19	71	18.15	40.33	15.62
20	277	73.71	155.20	53.3
Average	302	35.18	60.43	37.63

method, and Passive method to locate content sources. Since currently few useful sources can be obtained from the KAD network, we choose to not implement the KAD network function in the simulation. On the other hand, to evaluate the different impacts of stranger policies on system performance, we fully implemented the function of the credit system of aMule/eMule. As a result, general peers sharing different amount of content are assigned different credit values.

Simulation Setup

We introduced 10,000 client nodes and a single ED2K index server node into our simulation. The server node was always on-line, while the client nodes were randomly on-line or off-line. For these clients, 10 percent of them are free-riders, and the others are general peers. To simplify the simulation and concentrate on evaluating the stranger policies, we assigned each client node the same download

and upload bandwidth, 20KB/second. In the simulation, each client node must download a total of 48 files of the same size, 1000KB. The simulation time slot was defined as 100 seconds. At the beginning of each simulation, 100 general peers were already on-line in the system, and each of them randomly shared a different number of files, while the maximum numbers of shareable files were limited to 48. Every general peer performed content exchange processes according to the sequence: first discovering sources via different sources searching algorithms, and then both downloading from these sources and sharing files to other peers; while every free-rider did the same steps except for uploading their content. The peers finishing a file download might be continuously on-line to download the next file, or temporarily off-line with 2 time slots (200 seconds) on average, and then resuming the task again. This process kept running until the peers completed the whole task and chose to be permanently off-line.

To verify the prediction of our simulation model, we simply test two stranger policies for comparison. One is the original policy where strangers are given the maximum value of R_m: 10. The other one is the kindly restricting stranger policy where strangers are given the value of R_m: 5. Since we still keep the maximum value of R_m: 10, this setting will give less uploading bandwidth to strangers.

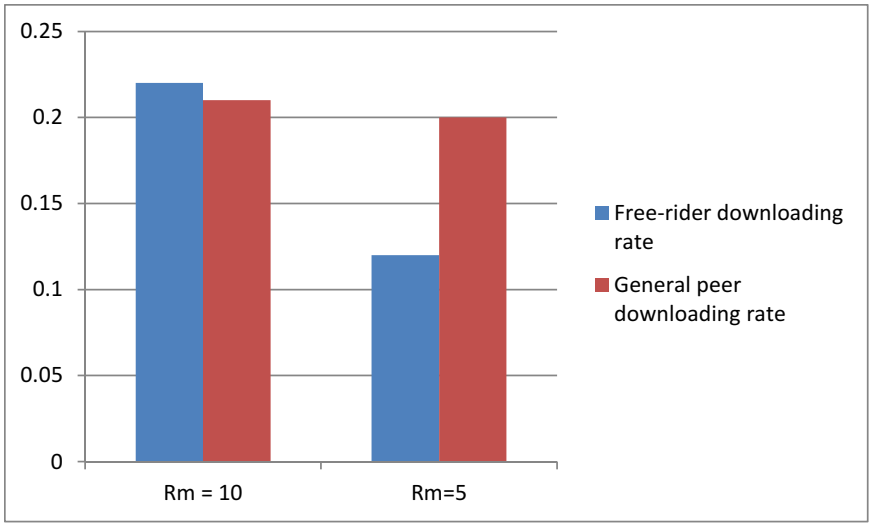


Figure 4.4.3: Simulation result showing a general peer's downloading rate vs. a free-rider's downloading rate

The simulation result is shown in Figure 4.4.3. With less uploading rate given

to strangers, the average downloading rate of general peers is not reduced too much, while the average downloading rate of free-riders has a noticeable decrease. Again, this simulation verifies our prediction that improving system fairness gradually will avoid the large reduction of system performance. However, simply reducing uploading bandwidth to strangers cannot entirely solve the free-riding issue, because free-riders can still obtain enough benefit from other design issues in the P2P file-sharing systems. For example, as we mentioned earlier, free-riders in the BitTorrent network can obtain similar downloading rate as general peers by collecting more downloading sources. Similarly, free-riders in aMule/eMule can also collect much more sources than general peers. To solve this issue, our previous work [45] proposed a novel free-riding control scheme for strengthening system fairness, and the details are shown in Appendix A.

In the real world, users may feel that the downloading rate in BitTorrent is higher than that in aMule/eMule, even though the performance is further emphasized in aMule/eMule. The reason is that system performance is also affected by other factors, e.g. the chunk size of a file and the efficiency of the searching downloading sources. A previous study in [71] pointed out that the overlarge of the chunk size (over 2MB) in P2P file-sharing networks will significantly reduce system performance. Correspondingly, the chunk size of a file is defined as 256KB in BitTorrent, while 9.28MB in aMule/eMule. So again, to develop a successful P2P file-sharing system, other appropriate strategies are essentially to be used in conjunction with an appropriate stranger policies.

4.5 Conclusion

In this chapter, we focuses on better understanding the challenge of designing stranger policies in a general P2P file-sharing system with an open environment. Current P2P systems use reciprocity-based incentive mechanisms to maintain system fairness. However, since free-riders/whitewashers cannot easily be distin-

guished from general newcomers, inappropriately treating strangers may reduce system performance or fairness. Therefore, the impact of a broad range of stranger policies from extremely rewarding strangers to extremely restricting them is evaluated under the indirect reciprocity incentive mechanism and the direct reciprocity incentive mechanism.

- In the case of indirect reciprocity incentive mechanisms being used, both an analytical model and an agent-based simulation are adopted for evaluation. Performance and fairness as functions of stranger policies are shown in Figure 4.3.6 and are characterized by specific behaviors. The highest performance of the system is obtained when both known peers and strangers are allocated some bandwidth, and accordingly free-riders will survive. Correspondingly, the best fairness is obtained when the extremely restricting stranger policy is used. Thus, highest performance and best fairness cannot be reached at the same time.

- In the case of the direct reciprocity incentive mechanism being used, an agent-based simulation model is developed. The simulation results reveal that the system performance and fairness have gradually change under different stranger policies except when extremely restricting strangers (Figure 4.4.1). When the extremely restricting stranger policy is applied, systems will obtain the best fairness but the lowest performance. When the extremely rewarding stranger policy is applied, systems almost achieve their highest performance. This simulation model also provides insight on different design emphases for dealing with strangers in BitTorrent and aMule/eMule. Through two case studies, we verify the prediction from our simulation results that strangers are extremely restricted in BitTorrent while extremely rewarded in aMule/eMule.

Chapter 5

Conclusions and Future Work

This dissertation focuses on several research challenges on P2P fairness. In this chapter, we summarize the main results obtained in studying these research problems, and then we suggest possible future work for additional research topics.

5.1 Conclusions

Maintaining fairness is necessary in P2P file-sharing systems. In this dissertation, we study P2P file-sharing fairness issues: the issue of maintaining fairness during the content information publishing/retrieving process and the issue of designing effective stranger policies on P2P fairness.

5.1.1 The Necessity of Fairness Maintenance During The Content Information Publishing/Retrieving Process

Exchanging content in P2P file-sharing systems includes two fundamental steps: the process of publishing/retrieving content information and the process of exchanging real content with other peers. While maintaining fairness for real content exchange has been widely studied, the necessity of maintaining fairness for the content information publishing/retrieving process has been overlooked.

In Chapter 3, through experiments in the KAD network, we attempt to analyze

the necessity of maintaining fairness during the content information lookup process in P2P networks. We conduct several large-scale measurements to analyze the maintenance of both routing and publishing tables which are the key components in the publishing/retrieving process. By deploying our distributed measurement framework on the PlanetLab testbed, we first test the availability and the similarity of peers' routing tables. Our results show that on average more than 80% of nodes in the routing table are online and less than 25% of records are the same among different routing tables of peers belonging to a searching tolerance zone. This means that the routing table is well maintained, and a peer can use it to find desired peers easily. After that, we measure the SLI publishing table and find that on average only around 25% of items in this table are online. Furthermore, our measurement reveals that over 75% of peers leave the system within one hour after publishing content. By analyzing the KAD protocol, we discover that the current maintenance method for the publishing table, the poorly designed incentive policy, and the selfishness of the publishing peer are the reasons for the low availability of the publishing table. Consequently, these reasons cause the poor lookup performance of the KAD network. Therefore, our experiments' results show that the system, designed for helping users publish and retrieve content information, could be almost useless, if it lacks an effective incentive policy. Based on our discoveries, we propose three possible schemes to improve the lookup performance.

5.1.2 Stranger Policies on P2P Fairness

Since strangers in P2P file-sharing systems can be either general peers or free-riders, when designing a stranger policy, restricting strangers is not necessarily better/worse than rewarding strangers for system performance and fairness. However, few quantitative evaluations for different stranger policies currently exist in literature.

In Chapter 4, we focus on better understanding the challenge of designing stranger policies in a general P2P file-sharing system with an open environment.

Current P2P systems either solely reward strangers or restrict them alternatively, which may reduce system performance or fairness. In the case of the indirect reciprocity incentive mechanism being used, we adopt both a numerical analysis and an agent-based simulation to evaluate the impact of a broad range of stranger policies from extremely rewarding strangers to extremely restricting them. We found the highest performance and the best fairness cannot be reached at the same time. The highest performance of the system is obtained when some bandwidth is allocated to both known peers and strangers, and accordingly, free-riders will survive. Correspondingly, the best fairness is obtained when the extremely restricting stranger policy is used. In the case of the direct reciprocity incentive mechanism being used, we design an agent-based simulation model and use it to reveal that the system performance and fairness changed gradually under different stranger policies except when extremely restricting strangers. We find that systems will obtain the best fairness, but the lowest performance when an extremely restricting stranger policy is applied. On the other hand, systems almost achieve their highest performance when an extremely rewarding stranger policy is applied. With this simulation model, we can also predict the different design emphases for dealing with strangers in BitTorrent and aMule/eMule, where strangers are extremely restricted in BitTorrent while extremely rewarded in aMule/eMule. These predictions are verified with experimental studies and agent-based simulations.

5.2 Future Work

In the future, the possible research topics in maintaining fairness of the content information publishing/retrieving process in the KAD network and on the design of stranger policies can be directed as follows.

5.2.1 Future work on studying fairness design in KAD network

According to several possible improvements discussed in Chapter 3 on overcoming low lookup efficiency in the KAD network, future work can be to study the impacts of these modifications on the KAD performance through quantitative analysis or simulation. Moreover, additional work has to been done on designing an efficient incentive policy for the KAD network. In order to achieve this task, designers may need to answer the following questions:

- which kind of incentive scheme (e.g. incenting generous behavior of peers through monetary payment or through reciprocity trading) should be employed?
- How can we effectively alleviate selfish behavior and prolong peers' staying time?
- To implement the fairness policy, should the fully distributed infrastructure of the KAD network be maintained? Or should some central control nodes be introduced into the system?
- Can the new designed fairness policy been easily implemented in the current KAD network?

5.2.2 Future Work on Studying Stranger Policies

Our studies covers the stranger policies under both the IRM and the DRM. However, there still exists a potential research direction: the stranger policies under regional incentive mechanisms. Regional incentive mechanisms have just been proposed recently in [38, 41], where the historical behavior of a peer could be spread within a limited scope. Since dealing with strangers bring additional complexity to fairness design, it is necessary to consider the possibility of distinguishing general strangers from free-riding strangers. For example,

- is the popular social network technology helpful in distinguishing strangers in a P2P file-sharing system?
- Which kind of technology is suitable for analyzing stranger policies under the regional incentive mechanisms?

Another possible research direction is to improve the accuracy of both our analytical and simulation model. For example,

- in the analysis model, we assume peers' arrival according to the Poisson distribution. In reality, the arrival pattern may also be affected by the content popularity. Thus, other possible distributions should be considered when modeling the arrival of peers.
- We also assume peers will stay into the system until they complete the entire task. However, due to some uncontrollable reasons, peers may permanently leave the system during the downloading process. Considering this situation will also bring more accuracy to our model.
- In the simulation model, the simulation ability of our method may be limited by the central node. However, this situation can be mitigated through distributed simulation methods. With the introduction of additional super nodes, the original simulation structure can be replaced by a new hierarchical distributed structure, so the scalability of simulation system will be expended and the delay of simulation will be reduced.

Bibliography

[1] Internet Study Retrieved Nov 12 2010 from, "http://www.ipoque.com/resources/internet-studies/internet-study-2008-2009."

[2] E. Adar and B. A. Huberman, "Free riding on gnutella," 2000. [Online]. Available: http://www.firstmonday.dk/issues/issue510.2000

[3] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *SIGecom Exch.*, vol. 5, pp. 41–50, 2005.

[4] D. S. Menasche, L. Massoulie, and D. Towsley, "Reciprocity and barter in peer-to-peer systems," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[5] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "Robust incentive techniques for peer-to-peer networks," in *Proceedings of the 5th ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2004, pp. 102–111.

[6] Gnutella Retrieved Dec 12, 2012, from, "http://http://en.wikipedia.org/wiki/gnutella."

[7] D. Hughes, G. Coulson, and J. Walkerdine, "Free riding on gnutella revisited: the bell tolls?" *IEEE Distributed Systems Online*, vol. 6, p. 1, 2005.

[8] R. Krishnan, M. D. Smith, Z. Tang, and R. Telang, "The impact of free-riding on peer-to-peer networks," in *Proc. 37th Annual Hawaii Int System Sciences Conf*, 2004.

[9] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free-riding and whitewashing in peer-to-peer systems," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 24, no. 5, pp. 1010–1019, 2006.

[10] L. Ramaswamy and L. Liu, "Free riding: A new challenge to peer-to-peer file sharing systems," *In Proceedings: HICSS*, 2003.

[11] R. Sherwood, S. Lee, and B. Bhattacharjee, "Cooperative peer groups in nice," *Computer Networks*, vol. 50, no. 4, pp. 523–544, 2006.

[12] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*. Budapest, Hungary: ACM, 2003, pp. 640–651.

[13] Z. Liu, H. Hu, Y. Liu, K. W. Ross, Y. Wang, and M. Mobius, "P2p trading in social networks: The value of staying connected," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[14] B. Q. Zhao, J. C. S. Lui, and D.-M. Chiu, "Analysis of adaptive incentive protocols for p2p networks," in *Proc. IEEE INFOCOM 2009*, 2009, pp. 325–333.

[15] aMule Officical Website Retrieved Feb 12, 2012, from, "http://www.amule.org."

[16] eMule Officical Website Retrieved Dec 12, 2012, from, "http://www.emule/edonkey-project.net/."

[17] B. Cohen, "Incentives build robustness in bittorrent," *1st Workshop on Economics of Peer-to-Peer Systems 2003*, 2003.

[18] M. Sirivianos, J. H. Park, R. Chen, and X. Yang, "Free-riding in bittorrent networks with the large view exploit," *IN IPTPS 2007*, 2007.

[19] T. Locher, D. Mysicka, S. Schmid, and R. Wattenhofer, "A peer activity study in edonkey and kad," in *International Workshop on Dynamic Networks: Algorithms and Security (DYNAS)*, 2009.

[20] S. Handurukande, A. Kermarrec, F. L. Fessant, L. Massoulie, and S. Patarin, "Peer sharing behaviour in the edonkey network, and implications for the design of server-less file sharing systems," *ACM SIGOPS Operating Systems Review*, vol. 40, p. 371, 2006.

[21] Y. Li, D. Gruenbacher, and C. Scogolio, "Only reward is not enough: Evaluating and improving the fairness policy of the p2p file sharing network emule/edonkey," *Journal of Peer-to-Peer Networking and Applications*, vol. 5, pp. 40–57, 2012.

[22] PlanetLab Retrieved Dec 12, 2012, from,, "http://www.planet-lab.org/."

[23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM Conference*, San Diego, California, August 2001.

[24] P. Maymounkov and D. MaziÃšres, "Kademlia: A peer-to-peer information system based on the xor metric," in *In Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[25] M. Steiner, T. En-Najjary, and E. W. Biersack, "A global view of kad," in *In ACM Internet Measurement Conference (IMC)*, 2007.

[26] eDonkey Retrieved Dec 12, 2012, from, "http://en.wikipedia.org/wiki/edonkey-network."

[27] C. Dellarocas, "Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior," *2nd ACM conference on Electronic commerce 2000*, 2000.

[28] H. Nishida and T. Nguyen, "A global contribution approach to maintain fairness in p2p networks," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 21, pp. 812–826, 2010.

[29] R. Brunner, "A performance evaluation of the kad-protocol," Master's thesis, Institute Eurecom, 2006.

[30] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*. London, UK: Springer-Verlag, 2001, pp. 329–350.

[31] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41–53, Jan. 2004.

[32] aMule Officical Website, "amule," http://amule.forumer.com/index.php.

[33] M. Steiner, E. W. Biersack, and T. Ennajjary, "Actively monitoring peers in kad," in *In Proceedings of the 6th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[34] D. Stutzbach and R. Rejaie, "Improving lookup performance over a widely-deployed dht," in *In 25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[35] J. Yu, C. Fang, J. Xu, E.-C. Chang, and Z. Li, "Id repetition in kad," in *IEEE International Conference on Peer-to-Peer Computing*, 2009.

[36] M. Steiner, D. Carra, and E. W. Biersack, "Faster content access in kad," in *in IEEE Peer-to-Peer Computing (IEEE P2P)*, 2008.

[37] G. Memon, R. Rejaie, Y. Guo, and D. Stutzbach, "Large-scale monitoring of dht traffic," in *In Proceedings of 8th International Workshop Workshop on Peer-to-Peer Systems (IPTPS)*, 2009.

[38] M. Steiner, T. En-najjary, and E. W. Biersack, "Exploiting kad: Possible uses and misuses," *ACM SIGCOMM CCR*, vol. 37, pp. 65–69, 2007.

[39] J. Douceur and J. S. Donath, "The sybil attack," *USENIX IPTPS*, 2002.

[40] Distributed Denial of Service Attack Retrieved Dec 12, 2012, from, "http://en.wikipedia.org/wiki/denial-of-service-attack."

[41] M. Steiner, T. En-Najjary, and E. W. Biersack, "Long term study of peer behavior in the kad dht," *IEEE/ACM TRANSACTIONS ON NETWORKING*, vol. 17, pp. 1371–1384, OCTOBER 2009.

[42] H. J. Kang, E. Chan-Tin, N. J. Hopper, and Y. Kim, "Why kad lookup fails," in *in IEEE Peer-to-Peer Computing (IEEE P2P)*, 2009.

[43] Python Offical Website Retrieved Dec 21, 2012, from, "http://www.python.org."

[44] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *In Internet Measurement Conference(IMC)*, 2006.

[45] Y. Li, D. Gruenbacher, and C. Scoglio, "Reward only is not enough: Evaluating and improving the fairness policy of the p2p file sharing network emule/edonkey," *Peer-to-Peer Networking and Applications*, vol. 5, pp. 40–57, 2011.

[46] M. R. Rahman, "A survey of incentive mechanisms in peer-to-peer systems," Cheriton School of Computer Science, University of Waterloo, Tech. Rep., 2009.

[47] Z. Ge, D. R. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-peer file sharing systems," *INFOCOM*, 2003.

[48] Zipf's Law Retrieved Dec 12, 2012, from, "http://en.wikipedia.org/wiki/zipfs-law."

[49] M. Lin, B. Fan, J. C. S. Lui, and D.-M. Chiu, "Stochastic analysis of file-swarming systems," *Perform. Eval.*, vol. 64, no. 9-12, pp. 856–875, 2007.

[50] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains Modeling and Performance Evaluation with Computer Science Applications.* Johe Wiley & sons, 1998.

[51] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *In ACM SIGCOMM.* Portland, Oregon, USA: ACM, 2004, pp. 367–378.

[52] Little's Law Retrieved Dec 12, 2012, from, "http://en.wikipedia.org/wiki/littles-law."

[53] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A performance study of bittorrent-like peer-to-peer systems," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 25, no. 1, pp. 155–169, 2007.

[54] Y. Tian, D. Wu, and K.-W. Ng, "Modeling, analysis and improvement for bittorrent-like file sharing networks," *IEEE INFOCOM 2006*, 2006.

[55] S. Petrovic and P. Brown, "Large scale analysis of the edonkey p2p file sharing system," in *In Processings of INFOCOM 2009, IEEE*, Rio de Janeiro, Apr. 2009.

[56] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, A. A. Hamra, and L. G. Erice, "Dissecting bittorrent: Five months in torrent's lifetime," *Passive and Active Network Measurement In Passive and Active Network Measurement (2004)*, pp. 1–11, 2004.

[57] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," *In 4th International Workshop on Peer-to-Peer Systems (IPTPS) 2005*, 2005.

[58] J. Yang, H. Ma, W. Song, J. Cui, and C. Zhou, "Crawling the edonkey network," in *Proc. Fifth Int. Conf. Grid and Cooperative Computing Workshops GCCW '06*, 2006, pp. 133–136.

[59] L. Plissonneau, J.-L. Costeux, and P. Brown, "Detailed analysis of edonkey transfers on adsl," in *Proc. 2nd Conf. Next Generation Internet Design and Engineering NGI '06*, 2006.

[60] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer, "Free riding in bittorrent is cheap," *IN HOTNETS 2006*, 2006.

[61] B. Fan, J. C. S. Lui, and D.-M. Chiu, "The design trade-offs of bittorrent-like file sharing protocols," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 2, pp. 365–376, 2009.

[62] Y. Li, D. Gruenbacher, and C. Scoglio, "Evaluating stranger policies in p2p file-sharing systems with reciprocity mechanisms," *Computer Network Journal, Elsevier*, vol. 4, pp. 1470–1485, 2012.

[63] NetLogo Retrieved Dec, 12, 2012 from, "http://ccl.northwestern.edu/netlogo/."

[64] F. C. Billari, T. Fent, A. Prskawetz, and J. Scheffran, "Agent-based computational modelling: An introduction," *In book: Agent-Based Computational Modelling - Applications in Demography, Social, Economic and Environmental Sciences*, pp. 1–16, 2006.

[65] M. Niazi and A. Hussain, "Agent-based tools for modeling and simulation of self-organization in peer-to-peer, ad hoc, and other complex networks," *Communications Magazine, IEEE*, vol. 47, pp. 166 – 173, 2009.

[66] M. Sasabe, N. Wakamiya, and M. Murata, "User selfishness vs. file availability in p2p file-sharing systems: Evolutionary game theoretic approach," *Peer-to-Peer Networking and Applications*, vol. 3, pp. 17–26, 2009.

[67] M. Yang, H. Chen, B. Y. Zhao, Y. Dai, and Z. Zhang, "Deployment of a large-scale peer-to-peer social network," *In Proceedings of Usenix workshop on real, large distributed systems (WORLDS).*, 2004.

[68] G. Swamynathan, K. C. Almeroth, and B. Y. Zhao, "The design of a reliable reputation system," *Electron Commer Res (2010) 10*, 2010.

[69] Y. Li, D. Gruenbacher, and C. Scoglio, "Understanding lookup performance deficiencies in the kad network," in *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, October 14-17, Pittsburgh, PA, USA*, 2012.

[70] Y. Tian and D. Wu, "Performance analysis and improvement for bittorrent-like file sharing systems," *Concurrency and Computation: Practice and Experience, CCPE Journal 2007*, Jun. 2007.

[71] P. Marciniak, N. Liogkas, A. Legout, and E. Kohler, "Small is not always beautiful," *In IPTPS' 2008*, 2008.

Appendix A

A new free-riding control scheme

To deal with free-riding behavior, we could require each resourceable peer adaptively adjusts the value of the modifier R_m assigned to its strangers, instead of directly giving the maximum value 10 to them. That is, give more uploading bandwidth to known peers. However, this strategy cannot entirely solve the free-riding issue, because free-riders can still obtain enough benefit from the other design issues in the P2P file-sharing systems. For example, as we mentioned earlier, free-riders in the BitTorrent network can obtain similar downloading rate as general peers by collecting more downloading sources. Similarly, free-riders in aMule/eMule can also collect much more sources than general peers. To solve this issue, we propose a novel free-riding control scheme for strengthening system fairness.

As we mentioned in the background section, two steps are necessary to complete the downloading task: one is to find the content sources at first, and then downloading the content from these sources. Since more download sources usually means higher download rate in P2P file sharing networks, the basic concept of our control scheme is to assign different number of sources to each peer according to its credit. Therefore, not only would contributors be rewarded with credit but also free-riders would be penalized by the restricted knowledge of available sources.

Our policy for allocating source information is based on the credit of peers.

After receiving sources requests, the general peer will return a different number of sources according to the credit that the requester has. The detailed policy is as follows.

- For the requester who contributed enough content to this general peer, i.e., the credit earned from this general peer is high, the general peer would provide all its known sources information. To judge whether the requester's contribution is enough, we can require the general peer to use a threshold like the average credit value of its former service providers as the judgment metric.

- For the requester whose credit value is lower than the threshold or without credit, the number of returned sources from this general peer would be proportional to the requester's credit. This rule addresses the situation of rate-limit contributions, where peers prefer to control their upload rate according to their download rate.

- For the strangers who has no documented data-exchange, i.e., it hasn't any credit, the general peer will return only very few sources to this free-riding suspect. Moreover, it also concurrently assigns a negative credit to such a suspect. The negative credit, which is stored by the general peer itself, will become positive if suspects show sharing behavior, or will become more negative as the suspect continues only to ask for sources. When the negative credit reaches a pre-defined value, e.g., negative three (the suspect has requested three times without contribution), the suspect will be judged as a free-rider and eligible for no more source information.

- Since we want to treat the general newcomer differently from a real free-rider (yet as the general newcomer has no shareable content, consequently it could be considered a free-rider), the general peer still provides sources rather than none to the stranger in the new scheme. However, the free-rider may avoid the punishment through the whitewashing attack where it

repeatedly change new identities[9], we require the general peer returns the same source information each time to every free-riding suspect, no matter what the suspect's ID is. Thus, different IDs can't benefit the free-rider any more. On the other hand, we admit that this scheme may delay the general newcomer's downloading progress, the limitation will disappear when the newcomer begins to share content.

Based on the main searching methods in aMule/eMule, our scheme can deal with the possible situation that free-riders may obtain enough sources through iteratively requesting their known resources. Under the Passive method, general peers agree to be sources of a requester only if this requester has their credit. Free-riders cannot obtain benefit from this Passive method. Under the Source-Exchange method, not only free-riders but also general peers will iteratively ask for more sources from their already known sources. Since our free-riding control scheme limits the number of returned peers to free-riding suspects (for example, only 10% back to suspects in contrast with 100% return to general peers), the number of sources obtained by general peers should be much larger than that obtained by free-riders. As a result, this could ensure that free-riders have less benefit than general peers. Furthermore, with the Source-Exchange method, we can naturally avoid the side-effect of the churn phenomenon, which may block a general peer when its sources are off-line. Instead of requesting a list of sources from one special peer, the Source-Exchange method helps peers actively and iteratively searches additional sources. Therefore, a general peer can still find out other online sources from its already known lists of sources who have assigned enough credit to it, even though it may be blocked by a special peer.

Of course, we can limit the total number of sources that free-riders can obtain in a special time period, but all those methods have their own advantages and disadvantages. For example:

1. instead of returning its known sources to the strangers at once, the requested peer will answer its strangers' requests after a special time interval. This can

delay the free-riders' ability to obtain sources quickly, and accordingly help the general peers join the uploaders uploading queue earlier than free-riders on average. However, it may also affect the general peers benefit when they are considered to be free-riding suspects;

2. to answer the requests from free-riding suspects, each requested peer can independently map the name of the requested file into an IP address format using a consistent map function. Then based on the knowledge of this same fundamental IP address, each requested peer selects its known sources whose IPs are among the closest to this IP address as their reply. This method can give free-riders almost the same sources no matter which nodes they request. However, it also increases the burden of those chosen sources;

3. each requested peer can return few sources to free-riding suspects, but this will postpone general newcomers potential contributing behavior, and accordingly reduces the system performance.

We admit that our scheme is a free-riding control method and not a free-riding removal method. Free-riders can still obtain information about some sources under this scheme. The reason is our scheme attempts to limit their benefit without modifying the infrastructure of aMule/eMule. Currently, the aMule/eMule is fully distributed and it has not any central control nodes. Moreover, the local knowledge of each peer is neither spread nor synchronized among other peers. This infrastructure makes the implementation of eMule/eDonkey simple and reduces the network traffic. Meanwhile, it cannot reliably fight selfish behavior or reward generous behavior, because neither free-riding nor contributing to one peer can be observed by other peers. To mitigate this issue, a global incentive policy can be an alternative of the local incentive policy, but it is more complicated even though it may be more efficient. Considering the success of aMule/eMule, our goal in this scheme is to keep its infrastructure unchanged and to try to find a trade-off between controlling free-riding and maintaining system performance. Consequently,

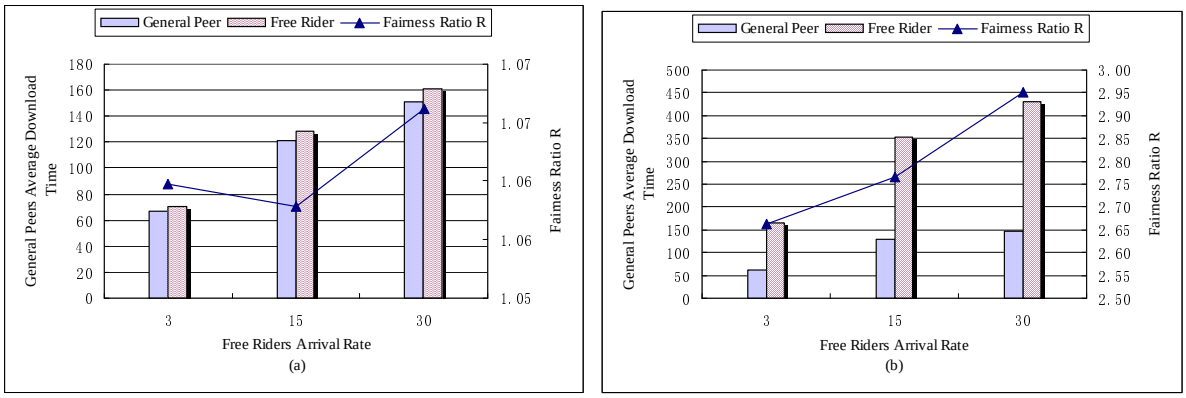


Figure A.0.1: Simulation results. The average download time of general peers and fairness ratio R with different values of control parameter β: (a) β =1; (b) $\beta = 0.1$.

this scheme also inherits the weaknesses of the local infrastructure of aMule/eMule, which does not allow fully removal of free-riders as well as restricting some general peers. However, this free-riding control policy remains simple, and the basic local structure of the credit system is unaffected, and the modification will be easy to implement in aMule/eMule.

To test the ability of the new scheme to deal with free-riding behavior, we also implemented the scheme into our agent-based simulation: first, the index server node will provide only a few initial sources to new coming peers and will no longer reply to repeated requests. Next, for the Sources Exchange method, we required the general peer to return all its known sources to peers with its credit, and to give the same small fraction of its known sources to peers without its credit. Again, remembering a suspect by a new additional field in the local database, peers will refuse to reply after being requested three times by such a suspect. For the Passive method, general peers agree to be the sources only for the peers having their credit. Even though we haven't implemented the KAD network in our simulation, the modification could easily be applied in the KAD method, which has the same operation pattern as the Sources Exchange method. To run the simulation, we required the general peer to return 10 percent of its total known sources to the free-riding suspects.

Our simulation results are shown in Figure. A.0.1. They indicates the validity

of our modification for controlling free-riding behavior, i.e., free-riders have to endure a significant delay in average download time in contrast to general peers.

Appendix B

Ping-pong message

The ping-pong message in aMule/eMule is used for testing the online status of peers. As Figure B.0.1 shown, peers in aMule/eMule can send a special hello-request message to other peers. If a hello-reply message can be received, the requested peer is proved online. The KAD protocol messages are transferred with UDP protocol. A special part of message called "opcode" is inserted after the UDP header. The hello-request message uses the opcode 0x10, and the hello-reply message uses the opcode 0x18.

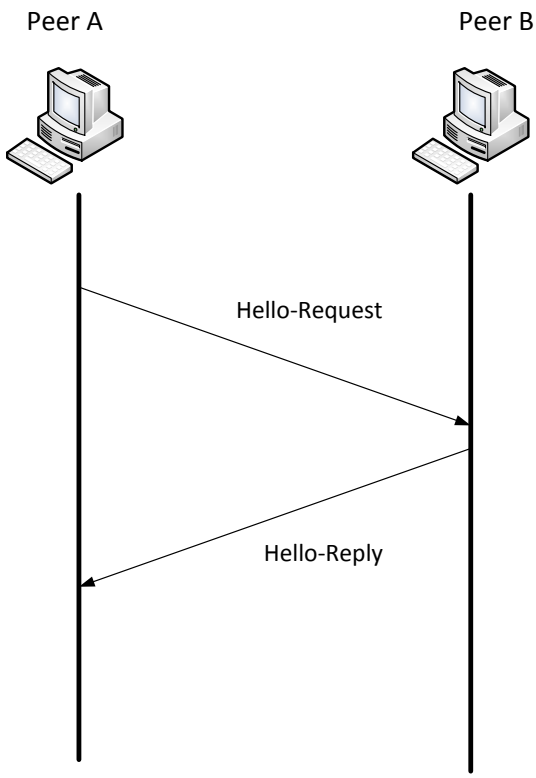


Figure B.0.1: Ping-pong message in the KAD network