COUNTERACTING FREE RIDING IN PURE PEER-TO-PEER NETWORKS

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
K. Murat KARAKAYA
March, 2008

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Dr. Özgür Ulusoy(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Asst. Prof. Dr. İbrahim Körpeoğlu(Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assoc. Prof. Dr. Ahmet Coşar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assoc. Prof. Dr. Nail Akar

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Asst. Prof. Dr. Ali Aydın Selçuk

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

COUNTERACTING FREE RIDING IN PURE PEER-TO-PEER NETWORKS

K. Murat KARAKAYA
Ph.D. in Computer Engineering
Supervisors: Prof. Dr. Özgür Ulusoy
Asst. Prof. Dr. İbrahim Körpeoğlu
March, 2008

The peer-to-peer (P2P) network paradigm has attracted a significant amount of interest as a popular and successful alternative to traditional client-server model for resource sharing and content distribution. However, researchers have observed the existence of high degrees of free riding in P2P networks which poses a serious threat to effectiveness and efficient operation of these networks, and hence to their future. Therefore, eliminating or reducing the impact of free riding on P2P networks has become an important issue to investigate and a considerable amount of research has been conducted on it.

In this thesis, we propose two novel solutions to reduce the adverse effects of free riding on P2P networks and to motivate peers to contribute to P2P networks. These solutions are also intended to lead to performance gains for contributing peers and to penalize free riders. As the first solution, we propose a distributed and localized scheme, called Detect and Punish Method (DPM), which depends on detection and punishment of free riders. Our second solution to the free riding problem is a connection-time protocol, called P2P Connection Management Protocol (PCMP), which is based on controlling and managing link establishments among peers according to their contributions.

To evaluate the proposed solutions and compare them with other alternatives, we developed a new P2P network simulator and conducted extensive simulation experiments. Our simulation results show that employing our solutions in a P2P network considerably reduces the adverse effects of free riding and improves the overall performance of the network. Furthermore, we observed that P2P networks utilizing the proposed solutions become more robust and scalable.

Keywords: Free riding, Peer-to-Peer networks, distributed computing, performance evaluation.

ÖZET

YAPISAL OLMAYAN EŞLER ARASI BİLGİSAYAR AĞLARINDA KATKISIZ KATILIMI ENGELLEME

K. Murat KARAKAYA
Bilgisayar Mühendisliği, Doktora
Tez Yöneticileri: Prof. Dr. Özgür Ulusoy
Yrd. Doç. Dr. İbrahim Körpeoğlu
Mart, 2008

Eşler arası bilgisayar ağları yaklaşımı kaynak paylaşımı ve içerik dağıtımında geleneksel istemci-sunumcu yaklaşımına karşı yaygın ve başarılı bir seçenek olarak oldukça dikkat çekmektedir. Ancak, araştırmacılar eşler arası bilgisayar ağlarının etkin ve verimli çalışmasını, dolayısıyla, bu yaklaşımın geleceğini ciddi olarak tehdit eden önemli miktarda "katkısız katılımı" bu ağlarda gözlemlemişlerdir. Bu nedenle, katkısız katılımın eşler arası bilgisayar ağları üzerindeki olumsuz etkisini azaltmak veya kaldırmak önemli bir araştırma konusu haline gelmiş ve bu alanda bir çok çalışma yapılmıştır.

Bu tezde, katkısız katılımın eşler arası bilgisayar ağları üzerindeki olumsuz etkisinin azaltılması ve kullanıcıların katkı yapmaya teşvik edilmesi maksadıyla iki yeni yaklaşım önerilmiştir. Bu ana yaklaşımlar, katkıda bulunan kullanıcıların başarımını artırırken katkısız kullanıcıları cezalandırmayı sağlayacak şekilde tasarlanmıştır. Birinci ana yaklaşımda, katkısız kullanıcıların tespiti ve cezandırılmasına dayanan dağıtık ve yerselleştirilmiş bir çözüm önerilmiştir. Bu yaklaşım, Bul ve Cezalandır Yöntemi olarak adlandırılmıştır. Eşler Arası Bağlantı Yönetim Protokolü adı verilen ikinci ana yaklaşımda ise, kullanıcılar arasındaki bağlantıları kullanıcıların katkısına göre yönetmeyi esas alan bağlantı tabanlı bir çözüm önerilmiştir.

Önerilen ana yaklaşımları değerlendirmek için yeni bir simülatör geliştirilmiş ve bir çok deney yapılmıştır. Simülasyon sonuçları göstermiştir ki önerilen ana yaklaşımların kullanılması, katkısız katılımın eşler arası bilgisayar ağları üzerindeki olumsuz etkisini azaltmış ve genelde başarımı artırmıştır. Bunlara ek olarak, önerilen ana yaklaşımları kullanan ağlar daha güçlü ve daha ölçeklenebilir hale gelmişlerdir.

Anahtar sözcükler: Bilgisayar ağlarının katkısız kullanımı, Eşler arası bilgisayar ağları, dağıtık hesaplama, başarım değerlendirme.

Biricik Anneme...

Acknowledgement

At the end of this study, I am very grateful that we have succeeded. I say "we", because this could not have been accomplished without the support of some people.

First of all, I am very grateful to my supervisors, Prof. Dr. Özgür Ulusoy and Asst. Prof. Dr. İbrahim Körpeoğlu for their invaluable support, guidance and motivation during my graduate study, and for encouraging me a lot in my academic life. Their vast experience and encouragement have been of great value during the entire study. It was a great pleasure for me to have a chance of working with them. I learned a lot from my supervisors, especially the endurance needed for this kind of study.

I would like to thank my thesis committee members Assoc. Prof. Dr. Ahmet Coşar, Assoc. Prof. Dr. Nail Akar, and Asst. Prof. Dr. Ali Aydın Selçuk for their constructive comments and suggestions for improving the manuscript.

I owe my warmest thanks to my colleagues Türker Yılmaz and İ. Sengör Altıngövde for their cooperation during this study. I would also like to thank my friends Latif Orhan, Murat Paşa Uysal, Ömer Faruk Gürel, and Ziya Yıldırım for their friendship and moral support. I have to express my gratitude to the Turkish Land Forces (KKK), Turkish Military Academy (KHO), and my superiors for supporting me during all these long years.

Above all, I am deeply thankful to my mother Saadet Karakaya and my sisters Selma Bayrak, Pervin Gonüllü and Hülya Çelebi along with my nephews, who supported me in each and every day. Without their everlasting love and encouragement, this thesis would have never been completed.

This work is partially supported by the Scientific and Research Council of Turkey (TÜBİTAK) under Project Codes EEEAG-104E028 and EEEAG-105E065 and with a Ph.D. scholarship.

Contents

List of Figures

List of Algorithms

List of Tables

List of Symbols and Abbreviations

DPM	:	Detect and Punish Method
OWRC	:	One-Way-Request Connection
P2P	:	Peer-to-Peer
PCMP	:	P2P Connection Management Protocol

Chapter 1

Introduction

The peer-to-peer (P2P) networking paradigm has attracted significant interest because of its capacity for resource sharing and content distribution. There are various architectures and applications of P2P networking, including file sharing, distributed computing, storage, collaboration, and multimedia streaming. In the ideal case, peers are expected to contribute to a P2P network by sharing their resources in turn of utilizing the network and the other peers' resources. However, it is observed that in many P2P networks, a considerable portion of peers are reluctant to share their resources [3, 46, 48, 93, 101]. Thus, the primary property of P2P networks, the implicit or explicit functional cooperation and resource contribution of peers, may fail and lead to a situation called *free riding*.

In P2P context, *free riding* means exploiting P2P network resources (through searching, downloading, or using services) without contributing to the network. A *free rider* is a peer that uses the P2P network services but does not contribute to the network or the other peers at an acceptable level. A *contributor*, on the other hand, is a peer that makes enough contribution to the network by sharing its resources with the other peers.

There may be various reasons and motivations for free riding. Bandwidth limitation of peers' connections may be one reason for free riding. Another reason for free riding can be the peers' concern of sharing "illegal" data on their own computers even though they are not concerned about using this type of data.

Some peers also have security concerns if they share something.

Researchers have observed the existence of high degrees of free riding in P2P networks, and they argue that free riding can become an important threat against the existence and efficient operation of P2P networks [3, 37]. As a result, a considerable amount of research has been done on free riding issue to diminish the impact of it on P2P networks.

In this dissertation, we propose two different solutions to deal with the free riding problem. These solutions aim to promote cooperation among peers and discourage free riding. As the first solution, we propose a distributed and localized framework which is based on detection and punishment of free riders. We call this framework Detect and Punish Method (DPM). Our second solution to the free riding problem is a connection-based framework, which we call P2P Connection Management Protocol (PCMP).

In DPM, we aim to design a framework which detects free riders and takes some counter actions against them. Thus, DPM consists of two separate mechanisms. The first mechanism is for detecting free riders by monitoring network traffic among one-hop neighboring peers. The second mechanism is for taking discouraging counter actions against the detected free riding peers. The mechanisms are distributed and localized. Basically, each peer is required to monitor its one-hop neighbors to decide if any of these peers is a free rider or not. Then the peer is required to take actions against the detected free riders.

The second framework, PCMP, introduces a novel P2P connection type, *One-Way-Request Connection* (OWRC) and a P2P connection management protocol that dynamically establishes the OWRCs between peers, and adaptively modifies the P2P topology in reaction to the observed contributions of peers. We designed PCMP based on the idea that if we can adjust the P2P network topology dynamically in reaction to peers' contributions, the adapted topology can favor the contributing peers in getting service from the P2P network. The adapted topology can also exclude free riders from the P2P network, and in this way the adverse effects of free riding can be reduced as well.

We implemented our solutions in a custom P2P network simulation tool that we

developed as part of this dissertation as well. Using our tool, we conducted extensive simulation experiments to evaluate our solutions and compare them against some alternatives. Our simulation results show that utilizing our frameworks leads to significant performance improvements for P2P networks. Furthermore, we observed that P2P networks employing the proposed free riding mechanisms become more robust and scalable.

1.1 Contributions

The contributions of this dissertation are as follows:

- A detailed survey of free riding in P2P networks conducted,
- A custom-designed pure P2P network simulation tool developed,
- A novel P2P network connection type and its management protocol proposed,
- A classification of observed free riding in P2P networks provided,
- Two novel frameworks against free riding designed, a detailed implementation of them in our simulator provided, and extensive simulation experiments performed to evaluate the frameworks,
- Impact of possible attacks and malicious acts against the implementation of the proposed frameworks evaluated.

The contributions presented in this dissertation have been published in two journals and a conference proceedings. Below is the list of these publications:

- M. Karakaya, İ. Körpeoğlu, and Ö. Ulusoy, "Counteracting Free Riding in Peer-to-Peer Networks", Computer Networks, Volume 52, Issue 3, February 2008.
- M. Karakaya, İ. Körpeoğlu, and Ö. Ulusoy, "A Connection Management Protocol for Promoting Cooperation in Peer-to-Peer Networks", Computer Communications, Volume 31, Issue 2, February 2008.

- M. Karakaya, İ. Körpeoğlu, and Ö. Ulusoy, "A Distributed and Measurement-Based Framework Against Free Riding in Peer-to-Peer Networks (short paper)", IEEE International Conference on Peer-to-Peer Computing (P2P'04), August 2004, Zurich, Switzerland.
- M. Karakaya, İ. Körpeoğlu, and Ö. Ulusoy, "GnuSim: A Gnutella Network Simulator", Technical Report BU-CE-0505, Department of Computer Engineering, Bilkent University, 2005.

1.2 Outline of the Dissertation

In the next chapter, we provide the background and related work for P2P networks and the free riding issue. In Chapter 3 and Chapter 4 we present our solutions to the free riding problem, DPM and PCMP respectively. The P2P simulation tool GNUSIM is presented in Chapter 5. In Chapter 6, we provide detailed results of our simulation study using GNUSIM for both solutions along with possible attacks to them. At the end of Chapter 6, we also compare the solutions and their performance. Finally, we conclude the dissertation in Chapter 7.

Chapter 2

Related Work and Background

Eliminating or reducing the impact of free riding on P2P networks has become an important research field in which a considerable amount of research has been done. In this chapter, we first have a discussion on classification of P2P networks, based on a variety of criteria. Then, we elaborate on the free riding problem in each class of P2P networks, along with the proposed solutions. Some possible attacks against these solutions are also discussed at the end of this chapter.

2.1 P2P Network Types

The impact of free riding and the effectiveness of a possible solution are related with the P2P network features and the provided P2P services. Therefore, before discussing the free riding issue further, we first would like to go briefly over various types of P2P networks in this section, and discuss how free riding can affect each of those in the next section.

P2P networks can be classified according to a variety of criteria [6, 68, 72] (see Table 2.1). One possible classification can be based on two features of networks; the "degree of centralization" and "degree of structure". The degree of centralization determines to what extent the P2P network relies on servers (none or some) to assist the interaction between peers, whereas the degree of structure refers to the way in which the content is indexed and located in the network. Using these

two criteria P2P networks can be classified into three types: *centralized*, *decentralized but structured* (hybrid), and *decentralized and unstructured* (pure). In *centralized P2P networks* there is a constantly-updated central directory which is used by peers to find out the location of resources. *Decentralized but structured P2P networks (hybrid)* do not have any central directory but they are structured, i.e., P2P network topology is firmly controlled and file indices are systematically placed at peers, following a certain algorithm. In this way queries can be resolved efficiently. In *decentralized and unstructured (pure) P2P networks*, there is no centralized directory and not much control over the network topology. The placement of file indices, if there is any, is not based on any knowledge of the topology and file indices are not related with each other. The most typical query method in such networks is flooding.

Criterion	P2P Network Types
Degree of centralization and structure	Centralized, Decentralized but structured (Hybrid), and Decentralized and unstructured (Pure).
Provided services	Distributed computing, P2P storage, File sharing, Collaboration, Platforms, Multimedia streaming, etc.
Legality of the shared content	All legal and Mostly illegal.

Table 2.1: P2P network types.

Another possible classification of P2P networks is with regards to the type of services provided by them, such as *distributed computing* (e.g., Avaki [9], Entropia [28], SETI@home [90]), *storage* (e.g., Freenet [33], Free Haven [34], OceanStore [77], PAST [78]), *file sharing* (e.g., BitTorrent [11], Gnutella [18], Napster [75], Publius [81]), *collaboration* (e.g., Jabber [50], Groove [38]), *platforms* (e.g., JXTA [56], MS .NET [74], the P2PTrusted Library [97]), and *multimedia streaming* (e.g., Freecast [32], Peercast [79], PPLive [80], UUSee [99]).

P2P networks can also be categorized according to the *legality* of the shared content in the network. For example, some P2P networks, such as official BitTorrent and renewed Napster services, are designed for distributing content on legal basis. However, there is a significant number of P2P networks which do not have any concern and mechanism for enforcing copyright. As a matter of fact, users of

these systems can abuse P2P network services to share pirated content illegally.

Since our solutions are based on decentralized and unstructured (pure) P2P networks, below we discuss their properties and mechanisms in detail.

2.1.1 Pure P2P Networks

In designing our solutions, we focus on pure P2P networks like Gnutella, because of their popularity and well-known open protocols [18]. Below, some of the distinct properties of pure P2P networks are summarized [1, 31, 88].

- There is no central coordination or central database.
- No peer has a global view of the system.
- Global behavior emerges from local interactions.
- All existing data and services should be accessible.
- Peers are autonomous and anonymous.
- Peers and connections are unreliable.

Some of these features enable pure P2P networks to be very successful, but some of them bring important problems. Among the problems of such networks is the so-called reputation problem. In a pure P2P network peers interact with unknown peers and have no information about their reputations. In other words, they do not know to what extent they can trust the other peers and the data provided by them. As a result, the detection of free rider peers and actions against them can not be easily implemented.

2.1.2 Phases in P2P communication

In a pure P2P network, a peer may go through four main phases which are implemented with descriptors in Gnutella Protocol [18] (See Table 2.2).

- Connection phase: A peer first finds some peers (from its cache, a central server, etc.) which have already connected to the P2P network. Then,

it requests connections from these peers by sending *Ping* messages. After receiving *Pong* messages, the peer sets up connections with these peers. Then, the peer can begin to communicate with the other peers in the P2P network.

- Search Phase: When a peer needs a file, it initiates the request by broadcasting the *Query* message to the P2P network through its neighbors. To limit the broadcasting of a *Query* message, Time-To-Live (TTL) value is included in the message header. The querying peer sets up TTL value to the maximum value defined by the P2P protocol.
- Downloading Phase: If the peer receives a *QueryHit* message, it begins to download the file from the source peer via a direct connection.
- Local Search and Routing Phase: Upon receiving a *Query* message via a neighbor, the peer first checks its local resources. If it has the file it returns a *QueryHit* message to the neighbor. No matter whether it has the file or not, it decreases the (TTL) value of the *Query* message by one. If the TTL value is greater than 1, the peer forwards the *Query* message to all neighbors other than the one which has delivered the search. If any *QueryHit* message arrives, the peer routes it back to the requesting neighbor.

A two-tiered P2P structure which divides peers into two groups (ultrapeers -or superpeers- and leaf peers) has also been proposed. Leaf nodes are located at the "edge" of the network and they are not responsible for any routing. The leaves are connected to the overlay through a few ultrapeers. On the other hand, the nodes which have high-bandwidth and are not behind firewalls are selected as ultrapeers. Ultrapeers accept leaf connections and route their queries. This approach reduces the number of messages forwarded towards leaf peers which in turn increases the scalability of the network. In this dissertation we focus on the flat pure P2P networks.

Descriptor	Description	Content
Ping	Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.	Nothing
Pong	The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.	IP and port of responding host, number and size of files shared
Query	The primary mechanism for searching the distributed network. A servent receiving a `Query` descriptor will respond with a `QueryHit` if a match is found against its local data set.	Minimum speed requirement of the responding host; search string
QueryHit	The response to a `Query`. This descriptor provides the recipient with enough information to acquire the data matching the corresponding `Query`.	IP and port, speed of responding host; number of matching files and their indexed result set
Push	A mechanism that allows a firewalled servent to contribute file-based data to the network.	Responding host id; file index;IP and port of requesting peer

Table 2.2: Gnutella Protocol Descriptors

2.2 Free Riding Problem in P2P Networks

The free riding problem is actually not unique to P2P systems. In the economics literature, the *tragedy of the commons* [47] is a similar problem with the free riding issue in P2P networks. *The tragedy of the commons* states the fact that selfish consumption of public goods may exhaust the whole public value. In this context, a public good can be defined as "a commodity for which use of a unit of the good by one user does not prevent its use by other users". Due to insufficient motivations to control individual behavior, people excessively consumes public goods, which leads to the tragedy of the commons problem. Over-fishing in deep oceans, pollution in cities, and over use of pesticides can be given as common examples of this problem.

In P2P networks, we can consider the services and digital objects as common goods because, for example, downloading a file does not prevent other peers from using it. As a P2P concept, *free riding* means exploiting P2P network resources (through searching, downloading objects, or using services) without contributing to the P2P network. A *free rider* is a peer that uses the P2P network services but does not contribute to the network at an acceptable level. A *contributor*, on the other hand, is a peer that contributes to the network by sharing its resources with other peers.

Various aspects of P2P networks have been investigated by many researchers. Some of the works on P2P networks have examined in detail the scalability, reliability, and workload issues [15, 39, 54]. Some researchers have analyzed the traffic and topology dynamics [39, 40, 84], while others have studied file popularity and availability in P2P networks [8, 17, 70, 94]. None of the works mentioned above, however, consider the free riding problem, its causes, or free rider demographics. The first study which specifically addressed the free riding problem in P2P networks was performed by Adar and Huberman [3].

Adar and Huberman extensively analyzed the peer traffic on the Gnutella network and they observed that 70% of peers do not share any files at all. Furthermore, 63% of the peers who share some files do not get any queries for these files. Another interesting observation is that 25% of the peers provide 99% of the all

query hits in the network. Having observed the existence of high degrees of free riding in P2P networks, the authors argue that free riding is an important threat against the existence and efficient operation of P2P networks.

Saroiu et al. confirmed that there is a lot of free riding in Gnutella as well as in Napster [92, 93]. They observed that 7% of the peers provide more files than all of the other peers combined. Moreover, Saroiu et al. compared the connection bandwidth reported by peers with the bandwidth calculated by direct observation, and found out that many peers misreport their bandwidth.

In a recent work [48] Hughes et al. pointed to an increasing downgrade in the network's overall performance due to free riding. Their results indicated an increasing level of free riding compared to Adar and Huberman's work. For example, they observed that 85 percent of peers share no files at all. They concluded that free riding is becoming more prevalent. The other findings of that work confirmed Adar and Huberman's overall findings. For example, they found that the top 25 percent of peers provide 98 percent of all query hits.

In another work, Yang et al. reported their findings about free riding in the Maze P2P system [101]. They also found a high level of free riding (about 80% of the peers). They observed that free riders were responsible for 51% of downloads, but for only 7.5% of uploads. These statistics suggest the existence of free riding in spite of the incentive mechanism provided by the Maze P2P system.

Recently, Handurukande et al. observed free riding in the eDonkey P2P network [46]. According to their findings approximatively 80% of the clients are free riders. Like the other research results mentioned above, most of the remaining clients share a small number of files. Less than 10% of the peers who are not free riders share considerable amount of files. As the authors concluded, the free riding phenomenon is common to most peer-to-peer file sharing systems, and the eDonkey P2P network is no exception.

It has been almost taken for granted that free riding is an unwelcome behavior and an important threat against the existence of P2P networks since the first observation. However, P2P networks succeed to survive in practice. Among possible reasons for this fact, altruism is of key importance. There are usually

altruistic peers in a P2P network, which can provide the required services, and the existence of them may enable P2P networks to survive despite free riders that exhibit selfish behavior [29]. The sense of being a member of a community, servicing other members, and gaining prestige among the others can be the motives for behaving altruistically [37, 53]. For example, SETI@home users share their computation power and bandwidth to detect intelligent life outside Earth without having a direct benefit. Other than altruism, peers can continue to share their resources by expecting that sharing resources helps to decrease the traffic at other peers from which they request some service [62]. Security concerns can be another important motive for some peers to stay obedient to P2P protocols. For instance, peers may still use an original client program that disables free riding instead of using a malicious version which enables free riding.

Even though generosity and altruism can play an important role in keeping on peer contribution in some P2P networks, not all P2P networks can depend solely on volunteer cooperation to achieve and maintain the desired level of service. In the absence of external motives, the amount and impact of free riding can exceed the acceptable levels depending on the requirements of different P2P networks. By employing free riding solutions, peers can be encouraged to contribute, negative effect of free riding can be diminished, and as a result, the aggregate utility of the network can be improved [62]. Therefore, eliminating or reducing the impact of free riding on P2P networks has become an important issue to investigate, and a considerable amount of research has been devoted to it.

2.2.1 Causes of Free Riding

There may be various possible reasons and motivations for free riding in P2P networks.

- Sharing resources is actually not free and may cost sharing peers in terms of bandwidth, hard-disk space, CPU cycles, etc. Therefore, a peer may want to avoid these costs by not sharing. For example, a peer may want to avoid the bandwidth cost of uploading. Many ISPs provide asymmetric connections which have relatively low uploading bandwidth. Therefore, peer's

bandwidth limitation and the network connections motivate free riding.

- If peers cooperation incurs some cost to themselves, and if the existing P2P protocol does not differentiate between free-riders and contributors, then peers do not have strong incentives to share. Since peers do not benefit from serving others, many peers decline to perform this altruistic act and become free-riders.
- Most of the P2P protocols are designed as if each peer were volunteered to cooperate and each peer contributes to the system equally, and thus they lack incentives and/or enforcements for sharing. Therefore, all peers enjoy the equal and same services even though some of them do not obey the expectations. If peers can use the P2P system and its resources for free and if they are not required to pay or to provide content in exchange of the service they get, then they may not be concerned about contributing to the system.
- Another reason for free riding can be the peers' concern of sharing copyright-infringing content from their own computers even though they are not concerned about using this type of content.
- Furthermore, some peers with a Network Address Translation (NAT) address act as a free rider even they do not intend to. Because, multiple computers share the same domain of IPs through NAT, and, if both peers are using NAT-based IP, they cannot download files from each other. These peers cannot upload files and therefore they would become free riders even they share files.

2.2.2 Impact of Free Riding

Free riding has some serious negative side effects on P2P networks as summarized in Table 2.3. In a free riding environment, a small number of peers serve a large population. Therefore, many download requests are directed towards a few serving peers, which may lead to scalability problems [82]. This also leads to a more client-server like paradigm [84, 92] and negates many advantages of the P2P

network structure. For example, the fault-tolerant properties of P2P networks may be weakened because a very small portion of the peers provides most of the content[1]. Renewal or presentation of interesting content may decrease in time; thus the number of shared files may become limited or may grow very slowly. The quality of searches process may degrade due to an increasing number of free riders in the search horizon. As the peers age in the network, they may begin not to find interesting files and may leave the system for good with all the files they shared earlier [39, 82]. Moreover, the large number of free riders and their queries will generate a large amount of P2P network traffic, which may lead to degradation of P2P services. Furthermore, underlying available network capacity and resources will be occupied by free riders, which will cause extra delay and congestion for non-P2P traffic as well.

Effect	Possible Consequences
A small number of peers serves a large number of requests.	Leads to more client-server like paradigm. Causes scalability problem. Weakens fault tolerance property.
Renewal and presentation of new content may decrease in time.	Satisfaction level of peers will decrease. Number of queries that will not receive any hit will increase.
Quality of search process may decrease.	Less number of hits will be returned. Satisfaction level of peers will decrease. Peers may stop using the system. Peer population may decrease.
Network traffic will increase.	P2P services may degrade. Delay, congestion, and loss will increase.

Table 2.3: Possible effects and consequences of free riding on P2P networks.

How serious is the effect of free riding on a P2P network depends on many factors including the P2P network type and its requirements (see Table 2.3). Since some resource types are not renewable, such as CPU cycles or disk space, it is very important what portion of peers are free riders in a P2P network that share those types of resources. For example, in P2P CPU-Sharing Grids, an example of *P2P distributed computing systems*, without sufficient level of CPU resource contribution, free riding can easily decrease the utility of the system or even can

[1] 1% of the peers provide 37% of the content [3].

collapse the system [4]. Similarly, in *P2P media streaming systems*, peers gain utility not only from the availability of files, but also from the ability to achieve high quality streams of these files [42]. The quality of a streaming session depends on a combination of factors, ranging from the characteristics of the streaming sources to the characteristics of the network paths. While a conventional file sharing system may be persistent with a low level of cooperation, a P2P streaming system cannot offer high streaming quality to its users if only a small portion of users cooperate [42]. Even though the network is not heavily congested, if the level of cooperation is low, the streaming quality would be poor [42]. Another type of P2P application that is very vulnerable to free riding is *P2P video multicasting systems*. In these networks, a piece of data (part of a video stream) arrives at a receiver over multiple hops of intermediate relaying peers. If an intermediate peer starts acting selfishly and refuses to relay data, the video stream will not arrive at any node in the sub-tree rooted at that free riding peer. Hence all nodes in that subtree of the multicast tree will not be able to receive the video stream. This is a fatal error for this application [73].

Structured P2P networks can be more vulnerable to some sorts of free riding than unstructured ones. In a structured P2P network that uses CAN (Content Addressable Network) protocol [83], for example, peers are responsible to store key-value pairs for keys that fall into their zone. A query in CAN is simply a key in the key space and its result is the corresponding value. A peer replies a query if the key is in its zone. Otherwise, it forwards the query to a neighbor. In the context of CAN, peers can also free ride by not storing key-value pairs in their zone and by ignoring incoming queries. This is a different type of free riding where a peer is not sharing an index either, not just the resource. If most of the peers free ride in this manner, CAN may easily fall apart and it can not resolve most of the queries [12].

The difference in P2P networks with regard to restrictions on sharing copyrighted content illegally plays an important role in free riding considerations as well. Most "illegal" content (pirated music, movies, books, etc.) sharing P2P applications do not care about free riding at all, since good P2P network performance and high user satisfaction are not that important for these networks. As the users of these

networks share copyrighted materials for almost free, they can bear degraded services. However, in "legal" content sharing, P2P applications care about their performance and user satisfactions.

As a result, free riding affects P2P networks in many ways and the level of impact may vary depending on the type of the P2P network and the application requirements. The effect may range from simply annoying the users to crashing the whole system. Therefore, a solution designed and implemented to deal with the free riding problem should be shaped according to the expected level of impact of free riding.

2.3 Securing Free Riding Solutions

Free riding and security problems should be studied together because solutions against free riding usually involve security mechanisms for protection from malicious acts [13]. However, deploying security mechanisms in P2P networks is quite difficult due to the characteristics of P2P paradigm such as anonymity, decentralization, self-organization and frequent disconnections [13].

Most security solutions used in networks of global scale require use of public keys for authentication, shared secret establishment, or integrity checking, and hence somehow depend on a public key infrastructure (PKI). Therefore we need to consider how PKI can be efficiently integrated into a P2P network. PKI is needed by asymmetric cryptography to establish the validity of the public keys. In asymmetric cryptography, a user needs two keys: a private key that is known only to the user, and a public key that is accessible to anyone. To authenticate the validity of the public keys, PKI stores digital certificates that attach a public key to the name of its owner by the digital signature of a trusted third party called the Certification Authority (CA). The management of certificates is a complex duty that requests a substantial infrastructure, especially in large-scale applications [13]. The services provided by the PKI cover up the whole life cycle of the certificates, including their issuance, distribution, suspension, and revocation.

In P2P context, direct implementation of PKI may be problematic. First of

all, pure P2P networks do not have any central management, which makes the standard PKI implementation based on CA hierarchy very difficult. Even in P2P networks with servers (hybrid or centralized), these servers usually do not fully control the peer behaviors as much as servers can do in a conventional client-server model. Thus, the centralized architecture of PKI may introduce several important problems that contradict with the important characteristics of the P2P networks [96]. One of the serious problems can be that the central servers and services may easily turn out to be the bottleneck of system performance, and thus the scalability of P2P network may become limited. For the network management, the realization of PKI entails a remarkable amount of resources to plan, install, deploy and maintain. For instance, PKI may need its own dedicated servers to function effectively. Furthermore, the huge number of users and high turn-overs in P2P networks make key management a challenge by itself. All these requirements hurt important characteristics of P2P paradigm by adding complexity. Reminding that specification document of the Gnutella protocol [18] version 0.4 is only 10 pages including the appendices, the complexity introduced by PKI would be understood better.

Another important issue of implementing security mechanisms is related with *anonymity* of peers which is one of the benefits of P2P networks provided to its users. Anonymity is related with hiding who performed a given action [13]. Providing anonymity, however, can open the doors for various security threats and malicious actions [96]. For instance, free riders can hide themselves or constantly change their online identities by exploiting anonymity mechanisms. A solution can be using a central trusted server, e.g., a CA, which can produce certificates for peer identification and supervise the validity of them. Rather than binding user identity to an arbitrary user information (an e-mail address, user name, etc.), these certificates can bind the identification to a public key. In this solution, new peers must connect to the CA before joining the network to get a certificate. However, if peer identities (for example IP addresses) are revealed peer anonymity is damaged to a certain extent. This means that user anonymity may be sacrificed to some extent for the sake of security.

A relevant concept to anonymity is *privacy*. If one can control when, where, and

how information about oneself is used and by whom, then it has the privacy [13]. To provide privacy, pseudonyms can be used to identify peers rather than their real identifiers [13]. The other peers in the system should not be able to link the pseudonym and the real identifier of a peer. Thus, pseudonyms can be used to refer to the subject that performed a given action without jeopardizing the privacy of that subject. However, in some of P2P networks, peers usually do not have a long-standing association with each other and with the network. As a consequence, user authentication depending on long-term secret keys, like in corporate networks [13], may not fit well. Therefore, in practice, a simple but less secure password-based user authentication has been extensively employed.

In summary, well-known client-server security solutions should be adapted for P2P paradigm to have robust and secure free riding solutions that can function in various P2P networks. Direct implementation of these solutions into P2P networks, however, may not fit the requirements and characteristics of P2P networks. In our solutions we do not require to use any kind of PKI implementations. Thus our solutions are free from the issues regarding security infrastructure which makes them practical and efficient.

The proposed solutions in this dissertation do not require any kind of extra security infrastructure, and, thus, they do not cause any significant overhead for securing them in the existing P2P networks. The data structures used in the proposed solutions are stored locally and there is no need to exchange information (score, utility value, reputation, etc.) about other peers in the network. However, malicious peer can still attack the solutions in various different ways. We discuss the possible attacks and how they can be dealt with in Chapter 6.

2.4 Approaches Proposed Against Free Riding

While cooperation is key to the existence and success of any P2P system, it is difficult to realize it without effective mechanisms. In fact, most of the implemented P2P systems lack such a mechanism and subsequently suffer from free

riding. Only a small portion of existing P2P systems have some mechanisms implemented against free riding, such as the ones described in [19, 26, 27, 71, 101]. To address this requirement, a number of approaches have been proposed to make P2P networks "contribution-aware" in order to combat free riding [5, 10, 12, 22, 25, 35, 37, 41, 42, 43, 44, 45, 58, 61, 62, 66, 69, 82, 95, 98, 102].

As the number of proposed solutions is quite large, we classified them into a number categories to aid the presentation and reading. This classification does not consist of an exhaustive list of all published work and does not imply that a single classification is possible. We put the solutions that have similar characteristics into the same category. There can be different ways of classification and naming of the categories. We tried to stick to the terminology which is already established in the literature.

The approaches proposed to deal with free riding problem can be categorized into three main groups (see Figure 2.1):

- *Micropayment-based Approaches:* These methods have been proposed to promote cooperation and discourage free riding within P2P networks by implementing micropayments.
- *Incentive-based Approaches:* These methods have been suggested as non-monetary mechanisms based on creating incentives for peers to share their resources.
- *Reputation-based Approaches:* These methods have been designed to create and distribute reputations of the peers by monitoring their past contributions.

2.4.1 Micropayment-based Approaches

In most of the P2P networks, the exchange of resources and services does not involve any monetary transaction. By providing efficient and secure pricing mechanisms, micropayment approaches are based on pricing peers for the services they get.

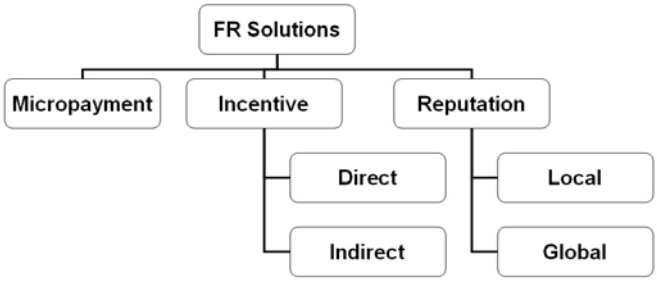

Figure 2.1: A classification of proposed solutions.

There are two key mechanisms in any micropayment system: an accounting module to securely store the virtual currency held by each peer, and a settlement module to fairly exchange virtual currency for services. The basic implementation of these components is to centralize their functions within a single central authority (a trusted third-party, a central bank, a broker, or a group of peers). This central authority manages each peer's balance and transactions by tracking accounts, distributing and cashing virtual currency. Most of the proposed solutions depend on a Public Key Infrastructure (PKI) for providing security against frauds and errors. As we discussed in Section 2.3, PKI implementation in P2P networks, however, has important issues. In essence, PKI has relatively heavy components which pose an additional burden on a P2P network [13].

As micropayment solutions deal with payments of small amounts, the incorporated security mechanisms should be quite lightweight [100]. Otherwise, the cost of the micropayment approach would overshadow the value of the payment. Therefore, most micropayment solutions do not guarantee totally fair exchange of goods and payment [100]. A tight security service would cause transactions to be more expensive (in terms of computation and communications) than the value of the exchanged goods. For example, in an off-line micropayment solution [100] coin fraud (analogy with using a counterfeit coin in a vending machine) may not be revealed until after the fact. However, offline payments may be preferred from a practical standpoint of performance improvements, such as lower latency, and lower communication and computational costs. This example shows the necessity of the question to what degree a network should be protected against malicious

and selfish peers. The reply to this question depends on the context of the network deployment and on the scale of the risk. Excess of protection can be harmful to the protection itself due to increasing complexity of the systems [13]. Effective micropayment systems simply require "good enough" security where fraud is detectable, traceable and unprofitable, while preserving high efficiency. A malicious peer should be avoided and disabled to continue using the services in the future.

Micropayment approaches are implemented using two different payment methods: online and offline. In online payment methods, the exchange of virtual currency takes place at the same time as the exchange of the services. This solution can prevent most of the payment frauds. To apply this method, the central authority must be reachable at the moment of transactions. On the other hand, in offline payment methods, the payment can be executed after the exchange of services if the central authority is not available at the moment. However, in offline payment methods, there are several important restrictions on the proposed systems, such as permanent identifications. Furthermore, because payments are offline, coin fraud (using a counterfeit coin) may not be discovered until after the fact. Still, offline payments might be preferred from a practical standpoint because they cause lower latency, and lower communication and computational costs.

Various micropayment approaches have been proposed in the context of P2P networks such as [35, 37, 44, 69, 71, 98, 100] among many others.

2.4.1.1 Implementation Issues

Micropayment-based approaches have several limitations when applied to P2P networks.

- *Centralization:* All proposed solutions require some centralized authority to monitor each peer's balance and transactions. However, this requirement conflicts with P2P paradigm, which is, by its nature, highly distributed. Furthermore, there is no simple way to decentralize micropayment approaches given that the central authority plays an important role in them.

- *Scalability:* Although payments could be online or offline, eventually the central authority must take some action for every transaction; as a result, the central authority's load is always directly proportional to the number of peers and transactions. It is clear that when scalability is of primary concern, a central authority constitutes both a bottleneck and a single point of failure.

- *Persistent identifiers:* To store peer balances and manage transactions, micropayment approaches require persistent user identifiers. Providing persistent identifiers, however, is complicated by the anonymity of peers, collections of widely dispersed peers, and the ease with which peers can modify their online identity in most of the unstructured and decentralized P2P networks.

- *Mental transaction costs:* Peers mostly dislike micropayments because of the fact that they have to decide before each download if the service is worth a few cents or not [44]. This leads to confusion and mental decision costs. Thus, micropayment solutions involve peers' mental effort in exchange for inexpensive resources, such as content, cycles, disk, etc.

- *Communication overhead:* There are two sources of communication overhead caused by introducing micropayments. The first overhead is created by dissemination of virtual currency value announcements, transaction records, etc. The second overhead is caused by the application of auditing mechanisms for integrity checking and expenditure monitoring.

2.4.2 Incentive-Based Approaches

In incentive-based approaches, P2P protocols promote cooperation among peers by providing some incentives. Service quality differentiation or prioritization of peers are common methods used by incentive-based approaches. In general, peers maintain histories of past behavior of other peers and use this information in their service differentiation decision. These approaches can be based on direct incentive (tit-for-tat) or indirect incentive (utility-based). In direct incentive approaches, a peer decides how to serve another peer based solely on the direct

service exchange between itself and this peer in the past. In contrast, in indirect incentive approaches, the decision of the peer also depends on the service that the other peer has provided not only to its neighbor but also to all peers in the system. Direct incentive approaches are appropriate for the networks where peers stay connected with long session durations, as they provide opportunities for creating a fair and realistic history of reciprocity between pairs of peers. Indirect incentive approaches are useful when the peer population is large and the chance of direct interaction with the same peer is low. The indirect incentive approaches provide faster information about a peer's past activities compared to direct incentive approaches.

Below, we provide more details about these two approaches.

2.4.2.1 Direct Incentive (Tit-for-Tat) Approaches

This kind of methods employs incentive mechanisms to encourage cooperative behavior between two or a set of peers. Each peer decides how to react to another peer's request depending on the past behavior of the other peer to its requests. Some existing P2P applications have implemented Tit-for-Tat approaches. For example, BitTorrent splits the original file into fragments [19]. To download all the fragments of a file, peers are required to exchange already downloaded fragments with the other downloading peers at the same time. In this way BitTorent employs a Tit-for-Tat approach by enforcing exchange of fragment among downloading peers. Additionally, the protocol increases the download speed of a peer if the peer provides more upload bandwidth.

The solutions that we propose in this dissertation implement a direct incentive mechanism. The Detect and Punish Method (DPM) is based on the local interaction of peers to create a direct incentive mechanism. Each peer assigns ratings to its neighbors based on the reaction of the neighbors to its service requests, and those ratings determine the service quality offered to the neighbors. In the P2P Connection Management Protocol (PCMP), we propose exploiting P2P network connection management as a direct incentive mechanism to promote contribution by reconnecting the contributors to each other and pushing the free riders away

from the contributors.

2.4.2.2 Indirect Incentive (Utility) based Approaches

These methods measure both a peer's contribution to the network and its resource consumption. This measure is termed the *utility* of the peer to the system which governs each peer's ability to consume network resources in the future. Utility-based approaches create incentives by providing better network services to the peers with higher utility. Peers with low utility value can face some form of penalty. For instance, they cannot download files or cannot even submit search requests if their utility value is less than the utility value of others or some threshold value.

As an example for indirect incentive-based approaches, in [60], the EigenTrust algorithm is used to measure a peer's contribution level to the P2P network by computing the peer's uptime, and the number, popularity and diversity of its shared files. The peers with high EigenTrust score are rewarded by better service quality, such as faster download or increased view of the network. Other examples of utility-based approaches against free riding include [42, 69].

2.4.2.3 Implementation Issues

There exist some critical issues to be considered regarding the realization of the incentive-based approaches.

- *Fake files:* A peer can share some small files with fake filenames resembling popular filenames. If these files are downloaded by others, this peer's utility value may increase.
- *Credibility of the utility value:* Some of the proposed incentive-based methods depend on accurate information about peers and this information is provided or stored by the peers themselves. A P2P network depending on such an approach can be cheated by writing malicious client programs.

- *Peer identity management:* Peers are linked with their utility value through their identities. However, a free rider can try to get rid of its reduced utility by whitewashing, i.e., by constantly getting a new identity, if newcomers are assigned a standard utility value which is higher than that of the free rider. Whitewashing issue is discussed in Section 2.5.

2.4.3 Reputation-Based Approaches

The goal of reputation systems is to allow peers to avoid dealing with peers who have bad reputations of being malicious or providing poor service in the past. These systems use the interactions among peers to build up a good reputation for contributing peers and a bad reputation for free riders.

In a reputation-based system, the information exchanged among peers can be positive reputations, negative reputations, or a combination of both. The systems that distribute only positive reputations take only the successful transactions into account to compute peer reputations. On the other hand, negative reputation-based systems share only negative feedbacks or complaints about peers. As a hybrid approach, a combination of positive and negative reputations can be disseminated and used in the network to make the reputation mechanism more accurate and reliable.

The reputation-based methods can be categorized into two main groups: *autonomous (local) reputation approaches*, in which peers use only their own experiences (local information), and *global reputation approaches*, in which peers use the experiences of other peers (global information) in evaluating peers.

2.4.3.1 Autonomous (Local) Reputation-Based Approaches

In an autonomous reputation scheme, a peer builds up local reputation information about other peers with which it has interacted by itself. Therefore, each peer can have different reputation values for the same peer. Unlike global reputation systems, autonomous reputation-based approaches do not aim to merge and distribute these local reputations to create a global consideration. As a result, they

are relatively simple to implement, because they do not call for a security infrastructure or centralized storage in order to assure the integrity of local reputations from other peers, unlike global reputation systems. Autonomous reputation approaches are used in some existing P2P networks such as eMule and GNUnet.

2.4.3.2 Global Reputation-Based Approaches

For a P2P network with a large peer population, any two peers may seldom or never interact. Therefore, it can take a long time to observe enough interaction between two peers to create useful reputations for their behavior toward each other. Global reputation-based approaches employ a reputation mechanism which depends not only on a peer's local interactions but also on other peers' interactions by consolidating all peers' local information. Various attacks can target at the reliability and integrity of global reputation information. Despite the security risks, global reputation approaches have the advantage of considerably speeding up identifying free riders, as peers can learn from others' interactions as well.

The reputation information can be distributed through the system in different ways. For example, in the XRep system [22], the reputation information that is locally created is stored at each peer, whereas in EigenRep [58], in addition to local reputation values stored at peers, the global reputation information derived from multiple local values is also stored at random peers. A peer retrieves any peer's reputation information from the system by using a retrieval mechanism.

2.4.3.3 Implementation Issues

Below we discuss some important issues that need to be considered when implementing reputation-based solutions.

- *Reliability:* Guaranteeing reliability and consistency of the reputation information gathered about peers is an important issue. There are a number of proposals against malicious acts such as using a voting scheme to collect opinions about a peer, implementing heuristics to find groups of potentially malicious voters, and applying a distributed cryptographic infrastructure

to confirm the identities of peers involved in a transaction.

- *Communication overhead:* In a global reputation system, peers need to communicate with each other or a special group of peers to exchange and consolidate reputation information, which increases P2P network traffic and can lead to scalability problems.

- *Complexity:* In a global reputation system, the need for ensuring the reliability of information received from other peers about their interactions with third parties can be met by adding security mechanisms to P2P network such as a cryptographic infrastructure like a PKI. A Certification Authority (CA) can be integrated into the P2P network to authenticate the reputation information being shared. This type of infrastructure might suit better to hybrid or centralized P2P networks, such as Napster or BitTorrent, than pure P2P networks, such as Gnutella. As discussed in Section 2.3, the implementation of PKI adds significant complexity to P2P management by entailing a remarkable amount of resources to plan, install, deploy, and maintain. Furthermore, the huge number of users and high turnovers in P2P networks make key management a complex issue.

- *Peer identity management:* Peers are linked with their reputations through their identities. Free riders can try to get rid of their bad reputations by constantly renewing their identities. Thus, P2P networks implementing reputation-based approaches should deal with identity management as well.

- *False recommendations:* Most global reputation systems assume that peers report their interactions with other peers honestly and impartially. However, a peer can cheat the system to benefit more at the cost of the others by misreporting the services received from other peers. If false recommendations can not be filtered out the fairness and effectiveness of a reputation-based approach will be jeopardized.

- *Centralization:* Global reputation systems may rely on a centralized authority to store and manage reputation ratings. Therefore monitoring peer reputations in a decentralized (pure) P2P network is problematic due to

the lack of a central authority. Furthermore, the required central infrastructure costs may be unreasonably high compared to the existing P2P infrastructure, and scalability of such a centralized system may be quite limited. For instance, it is argued that trust management in P2P networks does not scale well to many peers (i.e., when the number of peers is larger than 100,000) [12].

2.5 Common Attacks or Cheats

Some free rider peers could try to work around the free riding mechanisms if this would increase their benefits from the system. Solutions provided to prevent free riding should be robust enough against these kinds of attacks. Below we list some of the common attacks that can be mounted against the free riding solutions [10, 30, 44, 45, 58, 101]. These attacks are also summarized in Table 2.4.

- *Collusion:* A group of malicious peers can attempt to collectively challenge and fool the free riding mechanisms. For instance, a group of peers can collude to promote one or more peers in the group, or, to damage the reputation of a victim in a global reputation system. As another example, in some of the solutions against free riding, a peer can detect and announce a misbehaving peer. To evade being detected, cheaters may exploit these mechanisms by announcing an innocent peer or a potential announcer as a cheater.

- *Modifying virtual currency/utility/reputation value:* A cheater may exaggerate its virtual currency, utility, or reputation value by providing incorrect information about itself. Cheaters can do this by modifying client programs, cracking locally saved values, and so on.

- *Whitewashing:* In most current P2P networks, it is cost-free for a peer to join the network and obtain an online identity. This enables growing the network rapidly, since newcomers can easily join the system [29]. On the other hand, cheaters can use this fact to change their online identity anytime, and thus have all the advantages and rights of a newcomer. This

is called whitewashing. A free rider may choose to whitewash repeatedly to avoid being detected and getting punished. Incentive and reputation-based approaches are very prone to this kind of attack.

In many P2P networks, the real world identification of a peer is not bound to its online identity. Therefore, these systems can not easily locate a peer who has more than one account in the system or who enters the system repeatedly using a new identity each time. Distinguishing whitewashers from legitimate newcomers is an important issue to stop or restrict the cheaters.

A robust and long-term free riding solution should consider the possibility of these attacks and should incorporate mechanisms that can successfully deal with these kinds of attacks. For example, one technique that can be used against whitewashing is attaching a high cost to acquiring new identities for all newcomers using proof of work (POW) protocols [52]. As another measure against whitewashing attack, the free riding solution may require use of free but irreplaceable pseudonyms for peers through the assignment of strong identities by a trusted central authority [29]. An irreplaceable pseudonym for a peer can be, for example, the unique MAC (medium access control) address of the computer the peer is using. It is better that we consider the possible attacks as early as possible while designing a free riding solution, so that at the end we have a mechanism that can effectively deal with free riders and their workarounds.

Attack Type	Description
Collusion	A group of malicious peers arrange an attack and report incorrect information or promote each other.
Modifying Values	A cheater may exaggerate its virtual currency value.
Whitewashing	Cheaters change their identities and connections to erase their past records.

Table 2.4: Some common attacks to proposed solutions.

Chapter 3

Detect and Punish Method

In this chapter, we propose a new framework, the Detect and Punish Method (DPM), which is a distributed and localized solution against free riding in unstructured P2P networks. In DPM, peers' contribution to the network is monitored, and peers are enforced to act cooperatively in sharing network services and resources. The goal of this framework is not to eliminate all possible kinds of free riding. It is neither aimed to promote or enforce new content contribution by peers, as this may not be feasible. The aim of our low-overhead framework is to improve the current situation and reduce the ill-effects of free riding by detecting free riders, and reducing the amount of service they get from the network. In this way, peers are enforced to cooperate in order to use the services provided by a P2P network.

The benefits of DPM over the other mechanisms against free riding that have appeared in the literature can be summarized as below.

- In our work, we do not propose to use any scoring value for a peer's utility to the system. Thus, we do not need to bother with storing, retrieving, and saving a utility value. Each peer just stores information about the neighbors' messages which are routed through it.
- Unlike the other proposals against free riding, DPM does not require any permanent identification of peers or security infrastructures for maintaining a global reputation system.

- DPM does not require explicit cooperation of any group of peers to make the system work. Each peer executes the same kind of mechanisms alone and does not depend on any other peer's cooperation.
- As opposed to many solutions that execute the counter-actions at the download request phase, our solution executes some counter-actions at the query forwarding phase, i.e., during the search operation. In this way, our solution reduces not only the downloads performed by free riders, but also the query messages flowing in the network due to free riders. This considerably reduces the network traffic overhead.
- DPM requires minimal changes to the current protocol processing rules and it does not require any architecture changes.
- Both the detection mechanism and counter-actions are simple, practical and effective. Nor do they use large amount of resources.
- DPM categorizes the free riders into several categories. This enables us to apply several different counter-actions that are tailored to the types of free riding.
- DPM assesses the contribution of each individual neighbor to the monitoring peer and the overall system, on contrary to some other approaches which evaluate the contribution of the sub-network reachable via each neighbor.

In the following sections, we present the details about DPM, more specifically, the approach, the detection mechanism, the proposed free riding types, and the counter actions.

3.1 Main Approach

Our approach against free riding requires every peer to passively monitor its neighbors. Two roles are defined for each peer: *monitoring* and being *controlled* (see Figure 3.1). A peer takes both roles at the same time. As a monitoring peer, a peer monitors and records the number of messages coming from and going towards its neighbors (i.e., keeps some statistical information). The neighbors

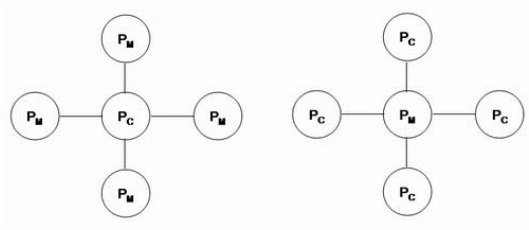

Figure 3.1: Peers are in two roles: monitoring and controlled.

are controlled peers. At the same time, the peer is also a controlled peer, which implies that its messages are monitored and recorded by its neighboring peers. By monitoring the messages of its neighbors, a monitoring peer can decide if a neighbor is acting like a free rider. Upon deciding that the neighbor is acting as a free rider, the monitoring peer can take counter-measures against that neighbor to reduce the adverse effects of free riding.

The statistical information[1] that a monitoring peer maintains about a controlled peer P consists of a set of counters that are shown in Table 3.1. These counters are maintained and updated by the monitoring peer as follows.

- QR_P, the number of `Query` messages `routed` by peer P, is incremented whenever the monitoring peer receive a Query message from peer P in which the TTL value is less than the fixed max TTL. The Queries originating from peer p are not counted; only the Queries originated at somewhere else and routed by peer P are counted. The monitoring peer decides if the Query originated by the neighbor or not by looking to the TTL value. If the neighbor P has originated the Query, then the Query message would have a TTL value equal to the fixed max TTL.

- QT_P, the number of `Query` messages routed `towards` peer P, is incremented whenever the monitoring peer sends a Query message to the neighbor P. Both the Query messages originated at the monitoring peer and the Query messages just forwarded by the monitoring peer are counted.

[1]Due to the power-law distribution of node degrees observed in P2P networks [55], we expect the average number of neighbors of a peer to be around 3-4, and therefore the overhead imposed by the solution on each peer will not be very large. Even the number of neighbors is larger than the average, the space and processing requirements are very low. This implies that the framework is scalable, thanks to its distributed nature.

Symbol	Description
QR_P	Number of `Query` messages `routed` by peer P.
QT_P	Number of `Query` messages routed `towards` peer P.
QH_P	Number of `QueryHit` messages submitted by peer P.
QHR_P	Number of `QueryHit` messages `routed` by peer P.
QHS_P	Number of `QueryHit` messages `satisfying` queries of peer P.

Table 3.1: Observed Descriptors.

- QH_P, the number of `QueryHit` messages submitted by peer P, is incremented whenever the monitoring peer receives a `QueryHit` message from peer P. The message must be originated (not forwarded) by peer P. The monitoring peer can decide this by checking the IP address field of the message, which stores the IP address of the originator of the message.

- QHR_P, the number of `QueryHit` messages `routed` by peer P, is incremented whenever the monitoring peer receives a `QueryHit` message from peer P in which the IP Address field in the message contains an IP address different than that of peer P.

- QHS_P, the number of `QueryHit` messages `satisfying` queries of peer P, is incremented whenever a Query message formerly submitted by peer P receives a `QueryHit`. To observe this, whenever a monitoring peer receives a `Query` message whose TTL is the fixed max TTL, it records in its internal table (using the *message ID* of the `Query` message) that the `Query` originated from the neighbor P. Then, after receiving a `QueryHit` message with the same message ID, the monitoring peer decides that the `QueryHit` message is for that controlled neighbor and increments the counter QHS_P. The monitoring peer counts only once for all the `QueryHit` messages received for the same query.

The values of these counters indicate both whether the neighbor is a free rider and the type of free riding. A different set of counters is maintained for each neighbor. The details of how we employ these counters are explained in the following sections.

We need to consider the issue of whether there is enough time during a typical

monitoring process to collect sufficient information about the neighbors to make correct decisions about their behavior. In one study [84], about 40% of peers in a Gnutella network leave the network in less than 4 hours; only 25% of the peers are alive for more than 24 hours. In another work [92], the average session duration of both Napster and Gnutella network clients is reported to be about 60 minutes. A similar work [39] found that 90% of Kazaa clients have sessions averaging 30 minutes in length. All these studies show that most peers in a P2P network stay connected long enough for monitoring peers to collect enough information to make correct decisions.

Another issue is whether a monitoring peer can monitor enough messages. In one study [70], the average number of queries received per second for three peers located at three different locations is about 50. In that same study, each peer received or sent an average of 30 query responses per second and the query response ratio per peer is around 10%-12%. This study shows that a monitoring peer will have enough messages forwarded through itself to or from a neighbor to judge if the neighbor is a free rider.

3.2 Free Riding Types and Detecting Free Riders

Previous works on free riding [4, 5, 37, 59, 98] have generally assumed that only one type of free riding is exhibited in a P2P network. However, studies [3, 39, 70, 82, 92] on P2P network traffic and user behavior suggest that not all free riders behave the same. Therefore, in this thesis we define three types of free riding (non-contributor, consumer, dropper) with different properties as summarized in Table 3.2. The types of free riding that we define here are not exhaustive. It is possible to define new types of free riding with different properties [61]. We believe that three types are sufficient for developing a general framework, and these free riding types that we focus on in this dissertation constitute a large fraction of all free riders. A detailed description of each type is given below.

Free Riding Type	NONE	NON-CONTR.	CONSUMER	DROPPER
Sharing Content?	Yes, much	No	Yes, but little	No
Replicating Content?	Yes	No	No	No
Routing Messages?	Yes	Yes	Yes	No
Request Gen. Rate	Normal	Normal	Higher	Normal

Table 3.2: Summary of free riding types and their properties.

3.2.1 Non-contributor

If a peer does not share anything at all or shares uninteresting files, it is identified as a non-contributor. A controlled peer P exhibiting this type of free riding can be detected by a monitoring peer who counts the `QueryHit` messages (QH_P) originating from the neighbor and compares them to the number of `Query` messages (QT_P) sent to the neighbor (Table 3.1) [2].

If the number of `QueryHit` messages received is very few compared to the number of `Query` messages sent, then the neighbor is identified as a non-contributor. More precisely, if the ratio (QH_P/QT_P) is below a threshold value, then the peer is identified as a non-contributor.

Not receiving (or receiving very few) `QueryHit` messages from a neighbor may indicate that the neighbor is either not sharing any files at all, or is sharing files that are not interesting and therefore they do not match the search queries. Unfortunately, a method like this, which is based on counting the `QueryHit` messages, cannot distinguish between these two types of reasons of not responding [3]. Different approaches for setting up a threshold value can be used [4]. Whatever the

[2] We can identify the source of a `QueryHit` message by looking at the `IP Address` field in the message, which stores the IP address of the responder.

[3] Peers who are cooperative but share unpopular files would be affected by false positives. From the perspective of the performance measures we have investigated, it seems that punishing such kind of users has a small impact on the overall performance of the network. A bias against these peers is one unintended consequence of emphasizing performance in an incentive mechanism. We acknowledge that the solution of this issue is beyond the scope of this thesis.

[4] We may, for example, set up a fixed value (say 100) for unsatisfied query number as a threshold. In this case, if $QT_P - QH_P$ is greater than this threshold, the neighbor is identified as non-contributor. As another approach, we may use a time-based threshold, such as 10 minutes, during which we monitor for `QueryHit` messages from the neighbor. If there is no `QueryHit` message received from the neighbor during this time period, the peer can be treated as a non-contributor.

approach, however, the proposed framework enables a monitoring peer to judge if a neighbor is a non-contributor just by observing the neighbor's existing protocol messages, without requiring that any new control message be defined for detection of free riders. Below, we formulate our method to detect non-contributors as a condition that is evaluated whenever an update is performed on the values of the respective counters. We have used this formula in our simulation experiments.

if $(QT_P > \tau_{QT})$ and $(\frac{QH_P}{QT_P} < \tau_{non\_contributor})$ **then**
peer P is considered as a *non-contributor*
endif

To eliminate the warm-up period and to obtain valid statistical information we propose using a threshold value, τ_{QT}, for the number of forwarded `Query` messages to the controlled peer. A monitoring peer starts making a decision about the controlled peer after this threshold is exceeded.

3.2.2 Consumer

Peers may contribute some content to the network. They are not therefore non-contributors, but the services they use may greatly exceed their contribution. This is not a desirable behavior in terms of the long term stability of the P2P network and fairness to other peers.

To identify whether a controlled peer P is a consumer, a monitoring peer counts the `QueryHit` messages that originate from the neighbor (QH_P) and the `QueryHit` messages that are destined to the neighbor (QHS_P). By comparing the ratio of these two values against a threshold value $\tau_{consumer}$, the monitoring peer can decide if the neighbor is a consumer or not.

In identifying consumers, the number of actual downloads, instead of QHS_P, could have been used. However, in unstructured P2P networks, the download process is executed directly between two peers [18]. Therefore, the intermediate nodes are not aware of the download process. This means that, the monitoring peers are not able to use actual download numbers to identify the consumers. Therefore, we propose using the `QueryHit` messages as an indication of possible

downloads. We assume that if a query gets one or more `QueryHits`, the owner of the query would download at least one copy of the requested file [5].

The following condition is checked to decide if a neighbor is a consumer or not whenever a respective counter maintained for the neighbor and used in the formula is modified. Again threshold for QT_P counter is used to eliminate the warm-up period before starting making decision about the behavior of a neighbor.

if $(QT_P > \tau_{QT})$ and $(\frac{QH_P}{QHS_P} < \tau_{consumer})$ **then**
peer P is considered as a consumer
endif

3.2.3 Dropper

A peer is identified as a dropper if the peer drops others' queries. Some peers might not forward protocol messages (`Query`, `QueryHit`, etc.) in order to save their connection bandwidth.

In order to detect a dropper peer P, a monitoring peer can count `Query` (QR_P) and `QueryHit` messages (QHR_P) forwarded by this neighbor. If the sum of these two values is very low compared to the number of `Query` messages sent toward the neighbor (QT_P), it can be assumed that either the neighbor does not have enough connections (to receive `Query` or `QueryHit` messages and forward them), or it drops `Query` and/or `QueryHit` messages. Again we can use a threshold value, $\tau_{dropper}$, for the ratio.

if $(QT_P > \tau_{QT})$ and $(\frac{QR_P+QHR_P}{QT_P} < \tau_{dropper})$ **then**
peer P is considered as a dropper
endif

[5] We only count once for all the `QueryHit` messages received for the same query. All `QueryHits` that is received for the same `Query` message will have the same unique message ID value.

3.3 Counter-Actions Against Free Riders

When a peer identifies a controlled peer as a free rider, the peer can start taking some actions against it. Here, we will focus on some sample counter-actions that can be implemented simply by modifying the existing P2P protocols.

Before discussing the details of counter actions, we would like to explain the relation between detection and counter acting. The detection of free riding and its type is determined according to the values of statistical counters maintained for a neighbor in the log table of a monitoring peer. When the values of the counters change, it may indicate that the type of free riding practiced by the neighbor has changed. For example, if the (QH_P/QT_P) ratio for a neighbor P is smaller than the respective threshold (i.e., the neighbor is a non-contributor), and later becomes greater than that threshold, the neighbor is no longer a non-contributor. Thus, monitoring peer would stop applying counter-action to the controlled peer since it is no longer a non-contributor. In essence, the counter-actions to peers are dynamic and changed according to the detection mechanism.

Our counter-actions are based on ignoring `Query` messages submitted by free riders or reducing the scope of these queries. In this way we reduce the amount of service that free riders get from the network. There are two main services that a peer can get from a P2P network: 1) *searching* for files by issuing `Query` messages; 2) *downloading* files after getting answers to the queries. If we reduce the amount of searching service that a free rider gets, we also cause a reduction in the amount of downloading service that it gets. Therefore, our counter-actions aim to reduce the propagation of `Query` messages submitted by free riders; then the free riders will have less chance of getting `QueryHit` messages and will perform fewer downloads.

We propose two types of counter-action schemes: 1) *single counter-action* schemes, and 2) *mixed counter-action schemes*. A single counter-action applies the same action to all types of free riders. A mixed counter-action scheme applies a different counter-action for each type of free riding.

The proposed single counter-actions are described in more detail below.

3.3.1 Modifying TTL Value

When a peer receives a `Query` message from a controlled peer, it first executes a search on local files for a match, and then forwards the `Query` to its other neighbors. Before the `Query` message is forwarded, its TTL value is normally decreased by one. However, the monitoring peer can play with this TTL value, i.e., the monitoring peer can decrement the TTL value by more than one before forwarding it further. In this way, the search horizon of the free-riding peer is narrowed. This also reduces the overhead imposed by `Query` messages on the network. To observe the effect of this counter-action at a finer granularity for different values of TTL reduction, we propose to employ two different values, i.e., 2 and 4, for decreasing TTL[6]. We call the corresponding counter-actions TTL-2 and TTL-4, respectively.

3.3.2 Dropping Requests

As a sharper counter-action, the monitoring peer can simply ignore all the search requests coming from a neighbor identified as a free rider. Dropping a `Query` message means not searching the local files for a match and not forwarding the `Query` any further; this is totally different from what happens in the Modifying TTL counter-action. We call this counter-action `DROP`.

Dropping the search requests of free riders or narrowing down their search horizon by modifying TTL not only punishes the free riders, but it also significantly decreases the overhead of P2P control messages over the underlying infrastructure. Uncontrolled query messages in a flooding-based P2P network can become a significant portion of overall network traffic[7]. We believe that decreasing the

[6] Actually, we implemented and observed the effects of different values between 2 and 6 in the simulation experiments. The results are provided in Section 6.1.4. We observed that TTL-2 has the least improvement effect on the performance and, TTL-6 and TTL-5 yield similar results to those of the DROP counter-action. TTL-4 produces a mid-point between them. Therefore, to give some insight about the effect of the Modifying TTL Action with different reduction amounts on the system performance, we select TTL-2 and TTL-4 as representative values in this thesis.

[7] For example, as it is pointed out in [85], 18 bytes of search string in a `Query` message may cause 90 megabytes of data to be forwarded by the peers of a P2P network. As another example, [2] states that the total number of messages including the responses triggered by a

number of queries submitted by free riders may help improve the performance and scalability of both P2P networks and the underlying Internet.

3.3.3 A Mixed Counter-Action

A monitoring peer that would like to execute a *mixed counter-action scheme* can apply an appropriate counter-action depending on the type of free riding. As mentioned earlier, a free-riding peer can be either a non-contributor, a dropper, or a consumer. Thus, a possible mixed counter-action scheme may dictate that counter-action TTL-2 is applied if the free rider is a consumer, counter-action TTL-4 is applied if the free rider is a non-contributor, and counter-action DROP is applied if the free rider is a dropper. In these settings, we aim to apply more severe counter-actions to free riding types that will cause more severe damage to the P2P network. A neighbor that is not identified as a free rider will not invite any counter-action.

3.4 Summary

In this chapter, we present the details of DPM by explaining the main approach, the proposed free riding types, the detection mechanism, and the counter actions. As a summary, DPM requires each peer to monitor the network traffic of its neighbors. By monitoring the messages of its neighbors, a peer can decide if a neighbor is acting like a free rider. If a peer determines that a neighbor is acting as a free rider, the peer can take counter-measures against that neighbor to reduce the adverse effects of free riding.

To evaluate DPM in a P2P network environment, we have conducted extensive simulation tests. In Chapter 6, we present and discuss the simulation results of the proposed solution in detail.

---

single `Query` message can be as large as 26240 (assuming 4 connections per peer).

Chapter 4

A New P2P Connection Management Protocol

Our second solution in this thesis, called P2P Connection Management Protocol (PCMP), is a connection-time solution that can be used to deal with free riding at connection establishment time. In this way it is quite different from the other methods that have appeared in the literature. PCMP is based on managing connections among peers to discourage free riding and to provide incentives for cooperation. PCMP involves the use of a novel connection type, One-Way-Request Connection (OWRC), and a P2P connection management protocol that dynamically establishes the connections between peers, and it adaptively modifies the P2P topology in reaction to the contributions of peers. Our claim is that if we can adjust the P2P network topology dynamically in reaction to peers' contributions, the adapted topology can favor the contributing peers in getting service from the P2P network. The adapted topology can also exclude the free riders from the P2P network, and thus the adverse effects of free riding can be reduced as well. Furthermore, it helps a P2P network to become more scalable and robust.

There exist some other studies which also focus on modifying P2P topology including the ones presented in [14, 16, 20, 67, 91]. However, they do not attack the free riding problem directly. These works basically aim either to solve the

topology mismatching problem for improving the search quality, or to modify the topology for decreasing overhead and increasing performance.

The boot-up and connection management mechanism in unstructured P2P networks allows a peer to join and leave a P2P network randomly, which causes topology mismatching between the P2P logical overlay network and the physical underlying network. The topology mismatching issue can cause a great amount of unnecessary congestion in the Internet infrastructure and seriously restrict the performance gain from various search or routing techniques. In [67], Liu et al. proposed a method called the Adaptive Overlay Topology Optimization (AOTO) to optimize inefficient overlay topologies for improving P2P search and routing efficiency. In another work [20], Crameret et al. also aimed to create a topology refinement by modifying the bootstrapping mechanism in the P2P network. Bootstrapping is an important core functionality required by every P2P overlay network. Peers intending to participate in such an overlay network initially have to find at least one other peer that is already connected to the network. The main idea of their approach is to use IP addresses to connect physically nearby peers together. For this, the proposed solution requires a peer that is part of the network to publish a reasonable (arbitrary k-bit) prefix of its IP address into a DHT (distributed hash table) directory. Then, a newly joining peer consults to this directory to find some physically close peers. In this way, peers try to establish connections with other peers that are physically close.

In [91], Singh and Haahr proposed to modify the P2P network topology so that peers with similar properties (bandwidth, geographic location, amount or type of shared resources, etc.) become close to each other. Similarly, in [14], Cai and Wang proposed a two-layer (neighbors and friends) unstructured P2P system for better keyword searches. The neighbors overlay is created according to network proximity while the friends overlay is built according to the online query activities. In order to increase the search quality, they try to avoid the free riders in the system while routing the queries. Primarily, the friends overlay is used to route the queries. Because, the friends overlay is constructed in such a way that free riders can not be friends of any peer. Thus the query would avoid free riders at the first place. In our proposal, we implement only a single overlay network

and our aim is to stop query submissions of free riders as well. However, in their system any peer, including free riders, may issue queries to the system which allows free riders to use the network resources. Actually, unlike our protocol, their proposal is not designed specifically against free riders. Chawathe et al. focused on scalability problem in unstructured P2P networks and applied dynamic topology adaptation [16]. They specifically aimed to match the query capacity of the peers with the routed queries to avoid the peers become overloaded by high query rates.

The properties of PCMP and its differences from the related previous works in the literature can be summarized as below.

- In our work, we do not propose to use any scoring value for a peer's utility to the system. Thus, we do not need bother with storing, retrieving, and saving a utility value.
- Unlike other proposals against free riding, PCMP does not require any permanent identification of peers or security infrastructures for maintaining a global reputation system.
- PCMP does not require the explicit cooperation of any group of peers to make the system work. Each peer executes the same kind of mechanisms alone and does not depend on any other peer's cooperation.
- PCMP is a dynamic approach. The connections are updated and changed dynamically depending on the interaction of peers.
- PCMP is autonomous. Each peer takes part in the connection management protocol autonomously (i.e., accepts a connection request by itself). No central management is required.
- PCMP depends on first hand observations. Decisions about connections are taken upon direct interaction with each peer. Therefore, the potential attacks by malicious peers are limited.
- PCMP is a simple protocol. The proposed method is very simple to implement and has a very small overhead for peers.

- The amount of change required to implement PCMP in the current unstructured P2P networks is not much, so the proposal is practical.

In the following sections, we first describe our main approach and highlight its benefits briefly. We then give the details of our two new connection types and the connection management protocol. We describe in detail how we establish and manage the connections between contributors and free riders. We also provide an example that shows how PCMP works and modifies a topology for the benefit of contributing peers.

4.1 Main Approach

P2P network topology affects the propagation of queries, the quality and quantity of search results, and the overhead imposed on the underlying physical network. Therefore, the connections among peers should be carefully controlled and managed. However, in current pure P2P networks, peers can try to connect to any other peer, and they can refuse any connection request to them. Each peer has equal right to do so, independent of their contribution level. Moreover, each peer can use all of its connections to send its queries. In our framework, we change these two properties of pure P2P network protocols to create an incentive for cooperation and to discourage free riding.

First, instead of a single connection type that exists in P2P networks to send and receive queries, we define the One-Way-Request Connection (OWRC) which introduces two new connection types: IN and OUT connections. IN connections are only used to receive queries and to reply them (i.e., provide service). OUT connections, on the other hand, are used just to send queries and to receive replies (i.e., request service). By using two types of connections, we can now differentiate and control service request and service provision separately.

Second, we propose a P2P Connection Management Protocol (PCMP) to establish and release these two types of connections. The protocol considers the peer contributions while establishing and releasing connections. Hence free riders can be disconnected from contributing peers and even get isolated sometimes. In

this way, the associated problems with free riding can be alleviated. Moreover, contributing peers may establish connections to not free riders, but to other contributors and therefore the number of contributors in their search horizon can be increased. Thus, contributors can have better chance to get query hits and downloads.

We foreseen several benefits of applying our protocol. The connectivity of free riders to the contributing peers can be reduced; in some situations, free riders can be totally isolated from the contributors. Furthermore, the connectivity among contributor peers can be increased. Also, the workload of a contributor peer can be reduced, since it will not serve many free riders anymore. As a result, better scalability and robustness can be achieved in the P2P network, since the querying overhead on contributor peers due to free riding can be reduced.

With those benefits, we can see improvement in terms of the following quantifiable metrics:

- Downloads for contributing peers can be increased;
- Downloads for free riders can be decreased;
- Amount of query traffic in the network can be reduced.

A motivational example and analysis about how PCMP can improve the performance in terms of some of these metrics in a P2P network is given in Appendix A.

An important issue in realizing our approach is to identify free riders efficiently and correctly. For this, we use a heuristic approach which depends on mutual exchanges of files and query hits between a pair of peers. Based on these exchanges, peers try to identify free riders and contributors. After then, they take necessary actions to modify their connections.

4.2 A New Connection Type: One-Way Request Connections

In the current pure P2P networks like Gnutella, a connection established between a pair of peers is used to exchange all types of P2P protocol messages in both directions including Queries, Query Hits, Pings and Pongs (Figure 4.1). PCMP modifies this assumption by proposing a new P2P connection type called *One-Way-Request Connection* (OWRC). As seen in Figure 4.2, an OWRC between two peers is still a TCP connection and can carry messages in both directions. However, there is a restriction on what types of messages can be carried in which direction of the connection. The connection is called one way because it can transfer requests in only one direction. In other words, over any OWRC the requests (Query, Ping) can only travel in one direction and the replies (Query Hit, Pong) can only travel in the other direction. Such a connection cannot be used to send and receive all kinds of protocol messages in both directions at the same time. The restrictions on the type of messages and their directions are enforced at the application level by PCMP.

In Figure 4.2, one end of the OWRC can be considered a *requester* (Peer A) and the other end as a *responder* (Peer B). The requester sends Query and Ping messages and receives the corresponding Pong and Query Hit messages via the OWRC. A responder, on the other hand, receives Query and Ping messages and replies with Query Hit and Pong messages through the same OWRC. In the rest of the thesis, we will call such an OWRC an *OUT-connection* at the requester end and an *IN-connection* at the responder end. Hence, in Figure 4.2, peer A has an OUT-connection and peer B has an IN-connection. We will also say that peer A has an *OUT-connected peer*, which is peer B. And peer B has an *IN-connected peer*, which is peer A.

If we would like to transfer requests from the other direction as well, from B to A, we need to establish another OWRC directed from B to A as depicted in Figure 4.3. However, we stress again that these connections are logical and can be implemented on top of either one or two TCP connections.

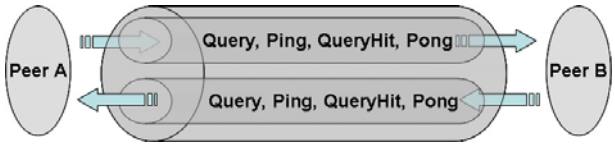

Figure 4.1: A general P2P connection between two peers, which enables both of them exchange all types of P2P messages.

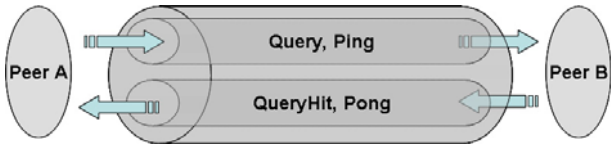

Figure 4.2: An OWRC between two peers, which limits the direction and the types of P2P messages exchangeable.

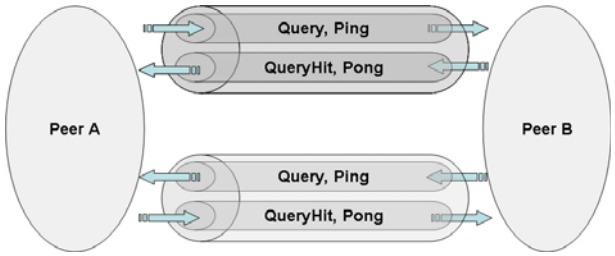

Figure 4.3: Two OWRCs between two peers, which enable each peer to request service from the other.

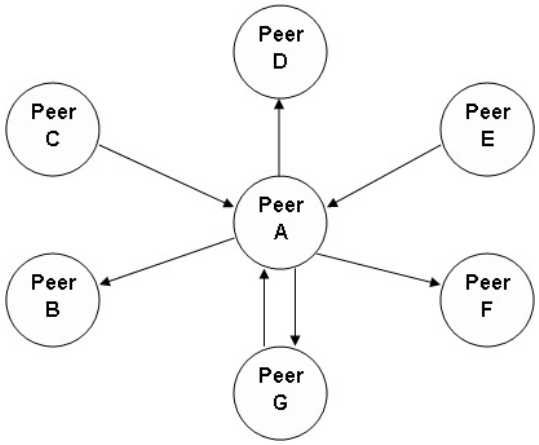

Figure 4.4: A directed graph representation of a network consisting of OWRCs.

A P2P network established using OWRCs can be modelled as a directed graph. A directed arc represents an OWRC: the tail of the arc has the peer that considers the connection as an OUT-connection, and the head of the arc (i.e., the pointing part) has the peer that considers the connection as an IN-connection. Hence the requests can flow along the direction of the arcs.

Figure 4.4 shows an example model of a P2P network consisting of OWRCs. Here, peer A has 6 neighbors. It has four OUT-connected neighbors (B, D, F, G) and three IN-connected neighbors (C, E, G). In other words, the IN-connections of A are {C, E, G}, and the OUT-connections of A are {B, D, F, G}. When Peer A would like to search the network it can submit the Query only to its OUT-connected neighbors, namely B, D, F, and G. It will process the Queries only coming from its IN-connected neighbors (C, E, G). If it receives any Query from OUT-connected neighbors it drops the request. The details of a peer interaction with PCMP are explained in Section 4.5.

We believe that peers would like to minimize the number of IN-connections, and they would like to maximize the number of OUT-connections. Because, IN-connections require a peer to process incoming Query and Ping messages, forwarding them and returning any replies to the originator. In contrast, more OUT-connections will help a peer to reach more peers and increase the probability of receiving a hit to its queries. In short, IN-connections require a peer to serve other peers, while OUT-connections allow a peer to use services offered by the

network.

4.3 Managing One-Way-Request Connections

PCMP manages OWRCs by taking the peers' contributions into account. Network topology adaptation as a result of PCMP actions aims to enable contributing peers discover each other more quickly and get connected to each other more directly. In this way, PCMP eventually results in topologies in which contributing peers are more closely located with respect to each other and free riders are more isolated.

Each peer executing PCMP can maintain zero or more IN-connections, and zero or more OUT-connections. Maximum number of IN- and OUT-connections is limited by the available bandwidth and determined by peers. The following data structures can be used to define an IN and OUT connection[1].

```
IN_Connection {
  long int PeerID;    /*ID of the other peer*/
  long int Downloads;   /*download counter*/
  double LastDwnldTime; /*last download time*/
}

OUT_Connection {
  long int PeerID;    /*ID of the other peer*/
  long int QueryHits;  /*Query Hit counter*/
  double LastQHitTime; /*last Query Hit time*/
}
```

According to PCMP, connections are updated at a peer whenever that peer is involved in a download or upload operation; otherwise, PCMP does not update

[1]Since node degrees in P2P networks follow a power-law distribution and average number of neighbors of a peer is observed to be around 3-4 [55, 68], we can argue that the overhead imposed by the solution on each peer will not be very large.

the connections of the peer[2]. The details of the PCMP operations that take place at requesting and providing peers are given below.

4.3.1 Managing IN-Connections

PCMP attempts to create an OWRC between the requesting peer (downloader) and the providing peer (uploader). The downloader will have an IN-connection from the uploader through which it can serve any future requests of the uploader. Since, the new OWRC is directed from the uploader to the downloader, it is an OUT-connection for the uploader on which the uploader can request service from the downloader.

The details of how an IN-connection is created by the downloader are given below.

- After the download, the downloader checks if there is an already created IN-connection coming from the uploader. If so, only the connection data structure is updated, i.e., the download counter is incremented by 1 and the last download time is set to the current time.
- If there is no existing IN-connection from the uploader to the downloader, a TCP connection is created between the downloader and the uploader[3]. The downloader waits for a Ping message from the uploader over the TCP connection. Because, after uploading, uploader is expected to request an IN-connection from downloader by sending a Ping message.
- If the downloader receives the expected Ping message from the uploader, it proceeds with the following steps:
  - If the downloader can accommodate a new IN-connection, it creates a new connection to the uploader. It then replies with a Pong message to the uploader. In addition, it creates an IN-connection structure, setting the download counter to 1 and the last download time to the

[2]Alternatively, the connections can be updated periodically rather than with every upload/download operation.

[3]This TCP connection will be used for PCMP's message exchange to create the new OWRC connection. If desired, the TCP connection used for file download can be used for this purpose as well.

current time.

– If there is no space to create a new IN-connection, connection replacement takes place. An existing IN-connection is replaced with the new IN-connection, i.e., the existing connection is released. The connection replacement policy is discussed in Section 4.4. Then, the downloader replies with a Pong message to the uploader. Again, the data structure for the connection is updated.

Algorithm 1 presents the pseudo-code for managing IN-connections.

Algorithm 1 Sample pseudo-code for managing IN-connections. A peer X will execute this code after downloading a file from peer Y. This pseudo-code is provided here to clarify the explanation in the text, and ignores some issues present in a real implementation. The code must be divided into several sub-functions, some of which can be executed asynchronously, as, when a Ping message arrives.

```
Download of a file F from peer Y has been finished;
InConn = Search for an IN_Connection to Peer Y;
if (InConn is FOUND) then
  /* update the connection structure */
  InConn.Downloads++;
  InConn.LastDwnldTime = now();
else
  Wait for a Ping message from Y;
  if (a Ping arrives from Y) then
    newInConn = Create_IN_Connection();
    newInConn.peerID = Y;
    newInConn.Downloads = 1;
    newInConn.LastDwnldTime = now();
    if (there is space in the IN_connection list) then
      Add(newInConn, IN_connections);
      Send a Pong message to Y;
    else
      victimInConn = SelectVictim(IN_Connections);
      Release(victimInConn);
      Add(newInConn, IN_connections);
      Send a Pong message to Y;
    end if
  end if
end if
```

4.3.2 Managing OUT-Connections

Upon uploading a file, PCMP attempts to create an OUT-connection from uploader to the downloader. If the connection is successfully established, the uploader can then use this new connection to send requests to downloader. The operations performed by the uploader to create an OUT-connection are described below.

- If there is an already-established OUT-connection at the peer to the downloader, the peer does not have to do anything, except possibly update some statistics.
- If there is no already-established OUT-connection to the downloader, the peer first creates a TCP connection to the downloader, through which further P2P messaging to create the OUT-connection can be done[4]. Then the uploader sends a Ping message to the downloader through this connection. Ping signifies that the uploader would like to establish an OWRC to the downloader. The downloader will consider the new OWRC an IN-connection, and it can either accept or reject the connection request. Normally, the downloader should accept the request if it obeys PCMP and if the downloaded file is not a fake file. The downloader will then send a Pong message back if it accepts the request.
- If a corresponding Pong message arrives from the downloader, the following operations are executed.
  - If the peer can accommodate a new OUT-connection, an OUT-connection to the downloader is created. The information about downloader is initialized: the downloader's ID is stored, Query Hit counter is set to zero, and the last Query Hit time is set to -1 (i.e., the value used when no Query Hit has been received yet).
  - If there is no space for a new OUT-connection, then the connection replacement policy is executed and one of the existing OUT-connections is replaced with the new connection.

[4]The existing TCP connection through which the upload has been performed can be used for this purpose as well, if we do not want to a create a new TCP connection.

According to the PCMP protocol, a peer sends query messages to OUT-connected peers through OUT-connections. If a Query Hit is received from an OUT-connected peer, the respective data structure for the OUT-connection is updated: the Query Hit counter is incremented by one, and the last Query Hit time is set to the current time.

Algorithm 2 shows the pseudo-code for managing OUT-connections.

Algorithm 2 Sample pseudo-code for managing OUT-connections. A peer Y will execute this code after uploading a file to peer X. This pseudo-code is provided here to clarify the explanation in the text and ignores some issues present in a real implementation The code must be divided into several sub-functions, some of which can be executed asynchronously, as, when a Pong message arrives.

```
Upload of a file F to a peer X has been finished;
OutConn = Search for an Out_Connection to Peer X;
if (OutConn is FOUND) then
  Update statistics;
else
  Send a Ping message to X;
  if (a Pong arrives from X) then
    newOutConn = Create_OUT_Connection();
    newOutConn.peerID = X;
    newOutConn.QueryHits = 0;
    newOutConn.LastQHitTime = -1;
    if (there is space in the OUT_connection list) then
      Add(newOutConn, OUT_Connections);
    else
      victimOutConn = SelectVictim(OUT_Connections);
      Release(victimOutConn);
      Add(newOutConn, OUT_Connections);
    end if
  end if
end if
```

4.4 Connection Replacement Policy

The connection replacement policy determines how to manage a limited number of IN and OUT-connections when all available connections of a peer are occupied and a new connection is required. There can be several different approaches for

designing replacement policies. In this dissertation, we propose two connection replacement policies. In the first policy, the number of downloads or the number of hit messages provided from the neighboring peer is employed to decide which connection to replace. The connection with the least number of downloads or hit messages provided is selected as a victim. We call the PCMP protocol employing this policy *Contribution-based PCMP (C-PCMP)*. In the second connection replacement policy, the time of the last download or the time of the last Query Hit provided from the neighboring peer is used to select the connection for replacement. The connection with the oldest time of the last download or hit messages provided is selected as a victim. We call the PCMP protocol that applies this policy *Time-based PCMP (T-PCMP)*.

4.5 A Peer's Actions and PCMP

Search: When a peer requires a file, it submits a Query through its OUT-connections.

Forward Queries: When a peer receives a Query from one of its IN-connections, it first searches its local files and replies according to whether the file was found. If the TTL value of the query is greater than 0, it forwards the Query through its OUT-connections.

Forward Query Hits: When a peer receives a Query Hit message from one of its OUT-connections and if the message is not destined to itself, the peer forwards the message towards the destination by using the IN-connection through which it has received the respective Query. The peer also updates the OUT-connected peer data accordingly.

Download: When a peer receives a Query Hit message from one of its OUT-connections as an answer to its Query, the peer requests the file from the uploading peer indicated in the Query Hit. A TCP connection is established between the peer and the uploader, and the download is started. Upon completion of the download, the peer receives a Ping message from the uploader; an IN-connection is created at the peer, and a Pong message is sent to the uploader as a reply to

the Ping.

Upload: When a peer receives a Query message through one of its IN-connections, it first searches its local files. If it can locate a matching file, it replies with a Query Hit message. Upon receiving the Query Hit, the Query originator requests the file from the peer. Upon completion of the upload, the peer sends a Ping message to the downloader to establish an OUT-connection towards that peer. Upon receiving a corresponding Pong message from the downloader, the OUT-connection is created and the peer can use it to send Queries.

4.6 PCMP Operation Example

As a simple example, consider the P2P network topology given in Figure 4.5. Assume each peer can only support up to 4 IN and 4 OUT-connections and the TTL is set to 2. The dashed circles represent the contributors (C1 and C2). In the given topology, the Query message of an indicated contributor (C1 or C2) cannot reach to the other one, since the indicated contributors are separated from each other by more than two hops. Assume a file F1 and a file F3 are stored on contributor C1, and a file F2 is stored on contributor C2. If the proposed PCMP is applied, the following scenario will occur.

- Peer P searches P2P network for file F1 with TTL 2. C1 replies with a Query Hit message. Then, Peer P downloads the file from the contributor peer C1. Upon download, Peer P deletes one of its IN-connections and adds a connection to C1 as a new IN-connection. C1 also removes (tears down) one of its OUT-connections and adds a connection to peer P as a new OUT-connection (see Figure 4.6).
- Then, contributor C1 searches for file F2 and the respective Query message reaches C2 via peer P. C2 replies with a Query Hit message, and C1 downloads the file from C2. After download, a new connection is set up from C2 to C1. It is an OUT-connection for C2 and an IN-connection for C1 (see Figure 4.7).
- Then, C2 searches for file F3, and C1 replies with a Hit message. After the

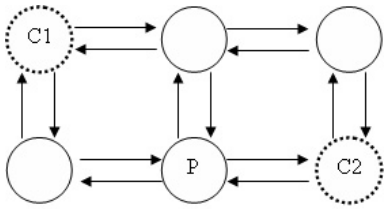

Figure 4.5: A sample topology layout.

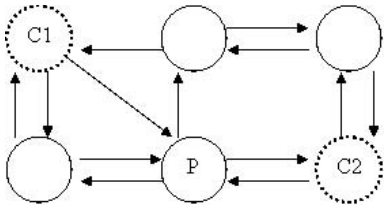

Figure 4.6: After download, Peer P updates its IN-connection by adding C1.

download has been finished, a new connection is established between C1 and C2. This time the connection is established from C1 to C2; hence it is an OUT-connection for C1 and an IN-connection for C2 (see Figure 4.8).

As seen in the above example, when PCMP is used, two contributing peers discover each other and get connected directly. Additionally, the free riders become further away from the contributing peers. If PCMP is not used, the two contributors could not benefit from each other; only free riders would benefit from this situation.

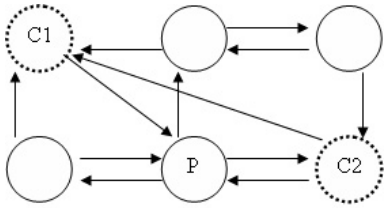

Figure 4.7: After download, Peer C1 updates its IN-connection by adding C2.

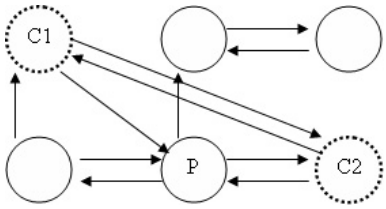

Figure 4.8: After download, Peer C2 updates its IN-connection by adding C1.

4.7 Summary

In this chapter, we present the details of PCMP by describing the main approach and highlighting its benefits. The details of two new connection types and a connection management protocol are also given. Briefly, we adjust the P2P network topology dynamically in reaction to peers' contributions. The adapted topology favors the contributing peers in getting service from the P2P network and restricts free riders' utilization from the P2P network.

To evaluate PCMP in a P2P network environment, extensive simulation tests have been conducted. The simulation results of the proposed solution are presented in Chapter 6.

Chapter 5

GNUSIM: A new P2P Network Simulator

In this chapter, we introduce a new P2P network simulation tool which we call *GNUSIM*. We implemented GNUSIM as an event-driven P2P network and protocol simulator using CSIM 18 [89] simulation library and C++ programming language on the WINDOWS OS. GNUSIM has been developed as a general purpose P2P network simulator, while it is used in this thesis specifically to:

- validate the proposed frameworks,
- measure and evaluate their performance,
- observe their effects on a P2P network, and
- compare the proposed frameworks and their variations.

The simulated P2P network model has many levels of detail such as the number of peers and files, network topology, content distribution, content replication, message handling, query pattern, query generation rate, free riding, and so on. Therefore, the model can be easily extended to simulate various types of P2P networks and protocols.

In the following sections we present the assumptions and simulation parameters associated with GNUSIM.

5.1 Assumptions and Parameters

The basic characteristics of the model are set to be similar to those of Gnutella network by implementing the protocol described in [18]. We present the main parameters of the simulation environment in Tables 5.1 and 5.2. Below, details of the important parameters and related assumptions are provided.

5.1.1 Network

Network Topology: The network topology defines the connectivity between peers. Any network topology configuration can be created by using a topology generator and then can be fed to GNUSIM. GNUSIM simply reads the topology from an input file. In performance tests, as a default setting we use a mesh structure to model the network topology (see Fig. 5.1).

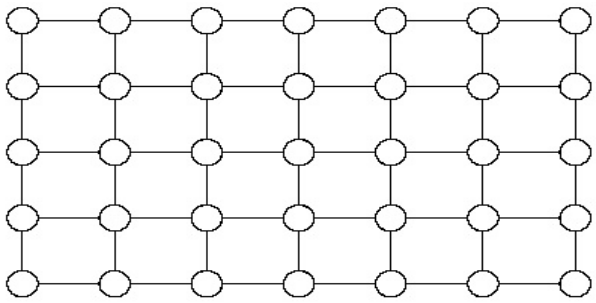

Figure 5.1: A mesh topology for network connections.

Messaging: There are two mailboxes within each peer. One mailbox is used for P2P network messages, and the other is used for TCP protocol messages, namely the download request and downloads. In this way, each mailbox simulates a port in Gnutella and TCP/IP protocol stack running on a peer.

Connection duration: A peer is supposed to stay connected in the network during the whole simulation lifetime.

Pinging Frequency: To check the validity of the connections with its neighbors, each peer submits a PING message at every PINGFREQ seconds.

Time-To-Live (TTL): In pure P2P networks, messages are broadcast into the network. The TTL parameter is used to limit the broadcast horizon in the networks.

5.1.2 Peers

Peers and Peer Types: We simulate a population of peers, consisting of both free riders and contributors. Peers are grouped based on the value assigned to the parameter which specifies the number of peer types and the corresponding properties that are provided in Table 5.1. We selected the default values for the peer type parameters in accordance with the observations provided in [3, 39, 70, 82, 92].

Ratio of Free Riders: At the beginning of the simulation, peers are grouped into different types based on the NUM_OF_PEER_TYPES parameter. The number of peers in each type is determined according to the POPULATION_RATIOS parameter considering the total number of peers (NUM_PEERS). For each peer type, we can set FREE_RIDING_TYPE to determine the free riding type of peers in that group. The values that can be assigned to FREE_RIDING_TYPEs are: NONE, NON_CONTRIBUTOR, CONSUMER, DROPPER, and MIXED. MIXED means that the peers in that type are equally and randomly distributed among all the free riding types defined. Peers' free riding types will not change during a single run of a simulation (i.e., during the simulation lifetime).

Download and Upload Bandwidth Capacity: We assume that each peer has a limited bandwidth capacity to download and upload files. Download capacity is assumed to be 1. There is only one download operation that can be executed at a time. However, a peer can upload more than one file at the same time, and the number of simultaneous uploads is limited by the NO_OF_MAXIMUM_UPLOADS parameter.

Download Attempts: If a peer reaches to NO_OF_MAXIMUM_UPLOADS, it can refuse more uploads. If a requesting peer is refused by a resource peer, it can try another source peer if there is any in the queryHitList. MAX_DL_ATTEMPT_NUMBER specifies how many times a peer should try to

Parameter	Definition	Default Value
NUM OF PEER TYPES	Number of peer types in the simulation.	3
POPULATION RATIOS	Population ratios of each peer type.	{0.10, 0.20, 0.70}
FREE RIDING TYPE	Free riding types of each peer type.	{NONE, NONE, MIXED}
SHARED FILE RATIOS	Ratio of the number of shared files of each peer type to total number of files in the simulation.	{0.87, 0.12, 0.1}
NO OF MAXIMUM UPLOADS	Maximum number of uploads a peer can provide at a given time.	{10, 10, 10}
QUERY GENERATION MEAN	The mean value of the exponential distribution that defines the time between two consecutive generated queries.	{60, 60, 60}
CONSUMER QUERY GENERATION MEAN	The mean value of the exponential distribution that defines the time between two consecutive queries generated by Consumer peers.	60
REPLICATION	If peers' REPLICATION property is set "true" then the downloaded files are replicated and shared.	{true, true, false}

Table 5.1: Peer Type Parameters

download the same file from different source peers if any peer refuses to upload the requested file.

5.1.3 Content

Content distribution: We distribute the content to peers uniformly and randomly. First, peers are grouped into different types based on the NUM_OF_PEER_TYPES parameter. Next, according to the given SHARED_FILE_RATIOS parameter, the number of files to be distributed for each peer type is calculated. Then, for each type of peers, the determined number of files are distributed uniformly and randomly. However, if a free rider peer is specified as a *dropper* or a *non-contributor*, no files are distributed to it. The files saved from this kind of peers are redistributed to *consumer* peers of the same peer type.

Content replication during simulation: The settings given for the content distribution above are valid at the beginning of the each run of the simulation. During the simulation the content distribution can be changed according to peers' property to replicate the downloaded files. If peers' REPLICATION property is set "true" then the downloaded files are replicated and shared after a successful download. Therefore, the content distribution in the system is dynamic during the simulation time.

Size of files: We assume that each file has the same size and the download time for all the files are the same which is specified by the DOWNLOAD_TIME parameter.

Uniqueness of the content: The number of distinct files (DISTINCT_FILES) and replication number of these files (COPY) determine the total number of files (TOTAL_FILES) to be distributed. For example if the number of distinct files (DISTINCT_FILES) is 100 and the COPY parameter is 2, it means that the total number of files (TOTAL_FILES) in the simulation would be 200.

Parameter	Definition	Default
MESSAGE PROCESSING TIME	Time to process any message.	0.1
MESSAGE WAITING TIMEOUT	The time duration of a peer to listen to its mailbox for any incoming message.	0.1
MESSAGE STORING TIMEOUT	The time for keeping information about a message: when timeout occurs all information about that message is deleted from routing table.	5.0
HIT WAIT TIME	The time for a peer to wait for the QUERY HIT messages arriving at itself before beginning to download process.	5.0
PINGFREQ	The time between two consecutive Ping messages.	20.0
SATISFIED QUERY HIT	Minimum number of QUERY HITS arrived to requesting peer to begin download process.	3
TIME TO LIVE	The maximum number of hops that a message can be transferred over.	3
MAX DL ATTEMPT NUMBER	The maximum number of attempts to download a file from arrived QUERY HITS.	3

Table 5.2: P2P Protocol Parameters

5.1.4 Request

Request-File Matching: We assume that the system replies the queries with exact matches only.

Request Pattern: Following a uniform distribution, peers randomly select a file to be requested from the P2P network. After selecting a file id to be requested, it is checked if the peer itself has the file. If the peer does not have the file, then it generates a Query message and submits it to all its neighbors.

Request Generation Rate: The inter arrival time distribution of requests follows exponential distribution with a mean value specified by the parameter QUERY_GENERATION_MEAN.

5.2 Summary

In this chapter, we present the details of our P2P network simulator, namely GNUSIM. We have developed GNUSIM as a general purpose P2P network simulator, which is used in this thesis specifically to validate and compare the proposed solutions. Using GNUSIM we have conducted extensive simulation tests to evaluate our solutions in a P2P network environment. In the next chapter, we provide detailed performance results for our frameworks.

Chapter 6

Experimental Results

In this chapter, we present and discuss the results of the simulation experiments for our two different frameworks, namely the Detect and Punish Method (DPM) and the P2P Connection Management Protocol (PCMP). At the end, we provide a discussion on the comparison of these two frameworks as well.

6.1 Simulation Results for the Detect and Punish Method (DPM)

Below we first present the assumptions and parameter values for the simulation experiments. Then, we explain the performance metrics observed in evaluating DPM. We then provide and discuss the results obtained in the simulation experiments in terms of the performance metrics. We also study some possible attacks to the proposed framework and their effects on the system.

6.1.1 Assumptions

Our model simulates a P2P network of 900 peer nodes. The peers are interconnected to form a mesh topology at the beginning of a simulation run. We assume that all peers stay connected in the same way until the end of a simulation

Property	Type A	Type B	Type C
Free riding type of the peers in the peer type.	NONE	NONE	MIXED
Population ratios of each peer type.	10%	20%	70%
Ratio of shared files of each peer type to total files.	87%	12%	1%
Peers replicate the files they downloaded.	True	True	False

Table 6.1: Properties of peer types.

run.

We assume that there are three types of peers in the simulated network: type A, type B, and type C. Type A and type B peers are contributors. Type C peers are free riders which can further be classified as non-contributors, droppers, or consumers. A type C peer is randomly and uniformly assigned to one of these 3 types of free riding. The properties of peer types are summarized in Table 6.1. The properties of each peer type include the population ratio, shared file ratio, maximum number of simultaneous uploads possible, query generation mean, and whether peers replicate the downloaded files or not. The default values of each of these properties are set similar to the values reported in [3, 46, 48, 92, 101].

At the beginning of each simulation run, peers are created according to the setup explained above and assigned to one of three main types (A,B, or C). During simulation, peers interact with other peers according to their assigned types. For example, a dropper drops all the messages, a non-contributor does not share any file, etc.

There are 9000 distinct files, with four copies of each, distributed to the peer nodes at the beginning of each simulation run. These 36000 files are uniformly distributed to peer groups according to the type of the groups and the file sharing ratios presented in Table 6.1. We do not distribute any files to peers that are free riders of the non-contributor or dropper type. We assume that each file is the same size and can be downloaded in 60 units of simulation time. During a simulation run, peers randomly select files to search and download, and they submit search queries for them. The inter-arrival time between search requests generated by a peer follows an exponential distribution with a mean of 60 time units. We assume that the query generation rate of consumer peers is twice that of other free-riding peers.

Each peer's upload capacity (the number of simultaneous uploads the peer can perform) is limited to 10. If a peer reaches upload capacity, a new upload request is rejected by the peer. The requesting peer can then try to download the file from another peer, selected from a list of peers obtained from the `QueryHit` message. We assume that the requesting peer repeats the same request a maximum of three times. After that, the peer gives up the downloading attempt and records this as an unsuccessful download. Then, it can initiate a request for another file.

Each simulation experiment is run for 2000 units of simulated time, repeated 10 times, and plotted on a 95% confidence interval.

6.1.2 Performance Metrics

In order to measure the performance improvement of DPM, we first determined a number of performance metrics. Below, we describe our metrics in detail.

- *Number of downloaded files*: This is an important metric indicating the number of downloads that can be performed in a P2P system during a fixed time interval. If peers can download more files from the P2P network, then level of satisfaction with the network will be higher.
- *Number of unsuccessful downloads*: The availability of content and services in a P2P network is an important issue. A network that is providing good service should not reject many of the contributing peers' requests. Since the network resources are limited, the upload capacity of peers contributing to the network will also be limited. If this limit is exceeded, the peers will start refusing download requests. If a peer can not be successful in downloading a file from up to three different source peers, it gives up downloading and records this downloading attempt as an Unsuccessful Download.
- *Number of uploads by contributors*: This metric indicates the load imposed on a peer. Contributors can become overloaded due to the excessive number of search and download operations they are involved in. Adapting free riding mechanisms in a P2P system, decreases the load on contributor peers by reducing requests from free riders.

- *Download cost*: We define the download cost for a peer as the ratio between the number of uploads and the number of downloads (i.e., #uploads/#downloads) performed by the peer. This ratio indicates the load imposed on a peer compared to the service the peer gets from the network. The smaller the ratio is, the better it is from the perspective of the peer.
- *Number of P2P network protocol messages*: This metric shows the messaging overhead in the P2P network and the underlying infrastructure. Messaging overhead affects the scalability of a system. In unstructured P2P networks particularly, the messaging overhead may be high due to the flooding approach used in querying. High numbers of protocol messages sent over the network also increase the level of congestion in the network. Congestion affects the performance of several network services in various ways, such as causing long delays for remote login applications, increasing query resolution time, and decreasing the speed of downloads [7].
- *Fairness*: Fairness metric shows that the level of service that can be used by a peer is proportional to the level of contribution that is provided by that peer. In other words, a peer contributing more than what is needed to overcome the thresholds is fairly compensated with more services. Thus, the solution encourages peers to contribute more and rewards peers based on the extent of their contributions.

6.1.3 Simulation Results and Analysis

In simulation experiments, we first tested the effectiveness of our detection mechanism in DPM. Afterwards, we conducted experiments to observe changes in the performance of a P2P network when counter-action schemes are applied.

6.1.3.1 Evaluation of Detection Mechanism

The detection mechanism is a crucial part of the framework. Therefore, we conducted extensive simulation experiments to measure the performance of our

framework in detecting free riders and free riding types. We used the following performance metrics to evaluate our detection mechanism:

- *Success ratio*: Ratio of peers correctly detected as free riders to peers designated as free riders in the beginning of each simulation run.
- *Sensitivity ratio*: Ratio of free riders whose free-riding type is correctly detected to the number of peers who have been detected correctly as free riders.
- *False alarm ratio*: Ratio of peers incorrectly detected as free riders to the number of peers detected as free riders.

A good detection mechanism should provide high values for success and sensitivity ratios and low value for false alarm ratio. The success ratio is an important metric for both single and mixed counter-action schemes; sensitivity ratio on the other hand is an important metric for mixed counter-action schemes, since in those schemes the type of free riding determines the counter-action to be applied. The false alarm ratio is a metric that indicates how many peers are incorrectly detected as free riders. If the false alarm ratio is high, it means that the framework applies counter-actions to contributors; thus some contributors are negatively affected by the incorporation of the framework into the P2P network.

An important restriction on the success of the detection mechanism is the behavior and ratio of droppers. This is because free riders of the dropper type usually can not use our detection mechanism, and hence can not apply any counter-action to their neighbors. As they do not route other peers' queries to their neighbors, they may not satisfy the detection mechanism's "routed query threshold (τ_{QT})" condition only by the count of their own queries. Therefore, in the overall detection results, droppers may play a negative role and limit the detection mechanism's success[1]. When the τ_{QT} threshold is decreased, however, droppers have more chance of satisfying the threshold value by recording only their queries, and they may then detect free riders. Thus, lowering the value of τ_{QT} increases the success ratio in the presence of droppers, as shown in Table 6.3.

[1]For example, in our simulations we observed that the peers about which droppers can not make any decision constitute around 20% of all the peers. This implies that our framework can not reach a success ratio better than 80% with the current settings of the simulation parameters.

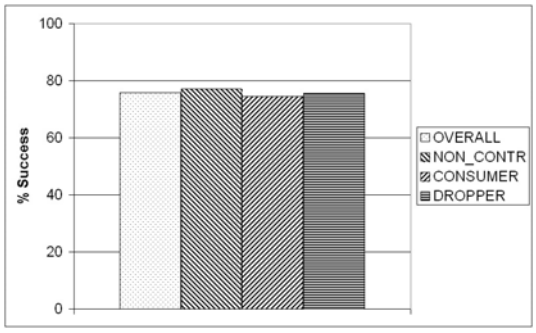

Figure 6.1: Success Ratio of detection mechanism in detecting free riders and identifying their free riding types.

Threshold	Description	Default	Range
τ_{QT}	Threshold value for the number of routed queries toward a controlled peer to begin the detection	50	25-100
$\tau_{non\_contributor}$	Threshold value for formula $\frac{QH_P}{QT_P}$ to decide if peer P is a non_contributor	0.001	0.1-0.0001
$\tau_{consumer}$	Threshold value for formula $\frac{QH_P}{QHS_P}$ to decide if peer P is a consumer	0.1	0.05-0.5
$\tau_{dropper}$	Threshold value for formula $\frac{QR_P+QHR_P}{QT_P}$ to decide if peer P is a dropper	0.1	0.05-0.5

Table 6.2: Threshold values for detection mechanism.

Figure 6.1 shows the Success Ratio of the detection mechanism for default values of simulation parameters. The overall success ratio is about 76%. This means that our detection mechanism is able to detect 76% of peers designated as free riders at the start of a simulation run. The false alarm ratio is about 9%. That is, 9% of the detected free riders were not really free riders. Their interactions with their neighbors during the simulations led the detection mechanism to identify them as free riders[2].

In Section 3.2, we use some threshold values for identifying each free riding type. Table 6.2 shows the default values of the thresholds. Default values are based on the P2P network traffic observations reported in [3, 39, 70, 84]. As part of our simulations we try to observe the effect of different threshold values.

[2]This level of false alarm ratio causes 9% of the peers detected as free riders to face counteractions; false alarms are a side effect of the detection mechanism. However, the performance metrics show that the performance is improved for contributors despite the false alarms (see Section 6.1.3.2).

τ_{QT}	Success	Sensitivity	False Alarm
25	95.39%	66.98%	13.82%
50	75.73%	66.84%	9.73%
100	75.38%	66.82%	9.70%

Table 6.3: Effect of τ_{QT} threshold values on the detection mechanism.

$\tau_{non\_contributor}$	Success	Sensitivity	False Alarm
0.1	76.54%	66.12%	42.87%
0.01	76.54%	66.12%	29.45%
0.001	75.73%	66.84%	9.73%
0.0001	73.03%	69.27%	5.24%

Table 6.4: Effect of $\tau_{non\_contributor}$ threshold values on the detection mechanism.

In Table 6.3, we observe that when the τ_{QT} threshold is set to lower values, the detection mechanism begins to detect earlier and the success ratio increases. However, the false alarm ratio also becomes worse with low values of τ_{QT} because the system tries to decide about a peer with less information available. There is therefore a trade-off between success and false alarm ratios and this trade-off is affected by the τ_{QT} threshold. Sensitivity is not greatly affected by the value of the τ_{QT} threshold.

Another threshold used in the detection mechanism is $\tau_{non\_contributor}$, which is used to decide if a peer is a non-contributor. Table 6.4 shows the effect of this threshold. Interestingly, for some large values such as 0.1 and 0.01 the success ratio does not change much, but the false alarm ratio changes and becomes too high. This result suggests that high values not be used for this threshold. The success ratio does not change much for different high values of the threshold, because even the precision of the ratio is different the number of detected peers with 0.01 is almost the same as with the value 0.1. That is, most of the non-contributor peers have a $\frac{QH_P}{QT_P}$ ratio less than 0.01. Therefore, the comparison leads to a similar success ratio. In Table 6.4, we again observe that the success ratio is (negatively) correlated with the false alarm ratio.

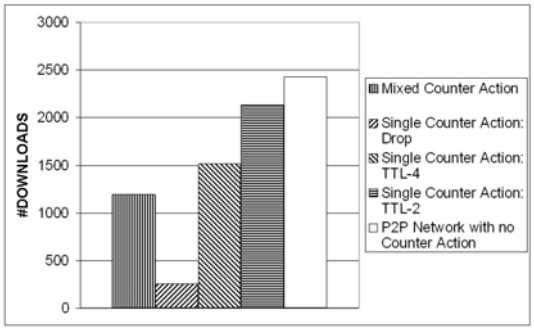

Figure 6.2: Decrease in free riding peers' downloads when different counter-actions are applied.

6.1.3.2 Evaluation of Counter-Actions

In Section 3.3 we proposed two types of counter-action schemes: single and mixed. We implemented three different single counter-action schemes: DROP, TTL-4, and TTL-2. We also implemented a mixed counter-action. This section evaluates the effectiveness of these schemes. The metrics we used in our evaluation are described in Section 6.1.2.

- *Downloads of free riders:* As Figure 6.2 shows, the number of downloads by free riders drops when mechanisms against free riding are applied. Counter-actions against free riders decrease the reach of the `Query` messages sent by peers detected as free riders; this reduces the chance of getting a hit to one of these queries. In this way, the average number of downloads by free riders is reduced. For example, the DROP counter-action causes a 89% reduction in the number of downloads by free riders. The least successful counter-action is the TTL-2 single counter-action, which achieves a 12% reduction. But even the least successful counter-action scheme leads to fewer free rider downloads than not using any counter-action at all.

  The success of the DROP counter-action is expected, since when all the queries submitted by free riders are dropped, those peers can not get `QueryHit` messages back, and therefore they can not download files. They can only download until they are detected. The other schemes are able to reduce the search horizon of the queries submitted by free riders, but the free riders still have the chance to get `QueryHit` messages and perform

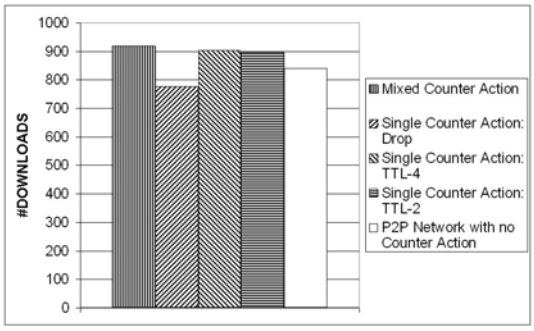

Figure 6.3: Increase in contributors' downloads when different counter-actions are applied.

downloads. The mixed counter-action scheme yields the second-best result. We believe that this approach has important consequences compared to single action schemes. Considering the potential false alarms that can be given by the detection mechanism, applying a different counter-action depending on the severity of free riding helps us to better deal with false alarms as discussed below.

- *Downloads of contributors:* It is desirable to increase the number of downloads for contributors. Since peers' upload capacity is limited, the download requests of contributors can sometimes be rejected. The rate of rejection is higher when there are many free riders in the system. Hence eliminating the effects of free riders on the P2P network will help to increase the number of downloads that contributors can make. This is indeed shown in Figure 6.3; applying our schemes achieves an increase in downloads performed by contributors as much as 10%.

  Figure 6.3 shows an important point; improvement in downloads is greater with a mixed counter-action compared to that with any single counter-action. While the mixed counter-action scheme produces about a 10% improvement, the two single counter-actions, TTL-2 and TTL-4, can produce about a 7% improvement. The DROP counter-action scheme actually reduces the number of downloads by contributors. We think this is due to false alarms in detection mechanisms. When we apply strict counter-actions such as DROP, the number of misdetected peers that are negatively affected

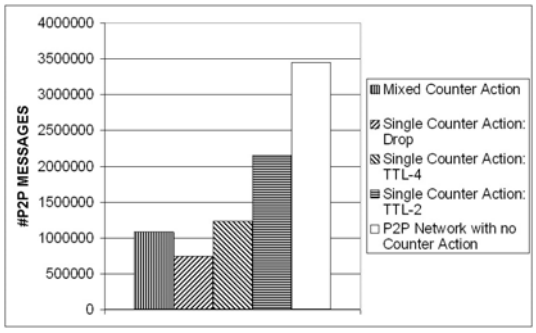

Figure 6.4: Decrease in P2P messages of free riding peers when different counter-actions are applied.

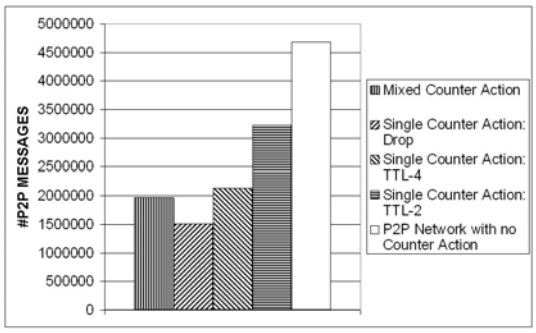

Figure 6.5: Decrease in P2P messages of all peers when different counter-actions are applied.

is significant. On the other hand, a mixed scheme handles false alarms better by applying different counter-actions to different types of free riders, and therefore can provide different levels of punishment, from light to severe, to peers suspected as free riders.

- *Amount of P2P protocol messages:* The number of P2P protocol messages transmitted in the network is an important factor affecting the scalability of the P2P network. Counter-actions against free riders result in up to 78% reduction in the number of transmitted P2P protocol messages (`Query` and `QueryHit`) originating from and destined to free riders (Figure 6.4).

  When we compare the reductions in transmitted P2P control messages for different counter-actions, we see that the DROP single counter-action again gives the best results (78%). The mixed counter-action scheme, on the other hand, reduces the control traffic due to free riders by about 68%.

If we evaluate the counter-actions with respect to their effect on reducing the total P2P control traffic in the network (i.e., the control traffic due to the free riders plus the contributors), we see that the DROP single counter-action scheme leads to a reduction of about 68%, whereas the mixed counter-action scheme leads to a reduction of about 58% (Figure 6.5). The least successful counter-action is TTL-2 which leads to a reduction of 31%. All these results show that applying the proposed framework helps a P2P network handle more peers with less control-messaging overhead and the system becomes more scalable with respect to the peer population.

The reduction observed in the number of protocol messages is the result of reducing or stopping the propagation of `Query` messages from free riders. As we restrict the propagation of `Query` messages by free riders, we also reduce `QueryHit` messages destined to free riders. The reduction of control traffic in a P2P network also means a reduction of traffic overhead imposed on the underlying infrastructure. This reduction translates to a better utilization of link bandwidths, and to a decreased processing load on the nodes constituting the underlying infrastructure.

- *Uploads of contributors:* A metric that can indicate the load on a peer is the number of uploads performed by the peer in a given time period. With our framework we want to achieve a reduction of the load on contributors. We expect that if we reduce the downloads of free riders, we can also reduce uploads, since a large portion of these uploads are done to free riders. In simulation experiments, we observed a significant reduction in the number of uploads done by contributors when a counter-action scheme is applied. As Figure 6.6 shows, the scheme that gives the best result is again the DROP single counter-action scheme, causing a reduction of about 68%. The mixed counter-action scheme causes a reduction of about 35%.

- *Download Cost:* The load on a contributor can also be defined in a different way as a normalized load, i.e., as the ratio of uploads to downloads. The results of our experiments show that our framework also causes a reduction in the download cost of contributors. As it can be derived from Figure 6.7, the framework achieves a 65% reduction in the contributors' download cost

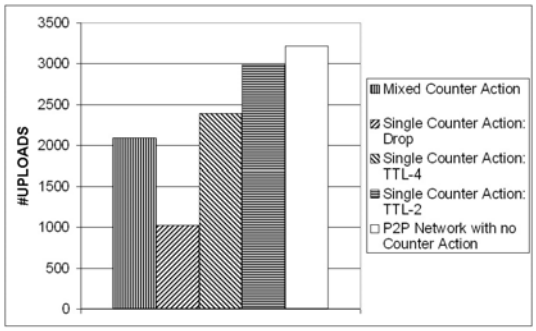

Figure 6.6: Decrease in contributors' uploads when counter-actions are applied.

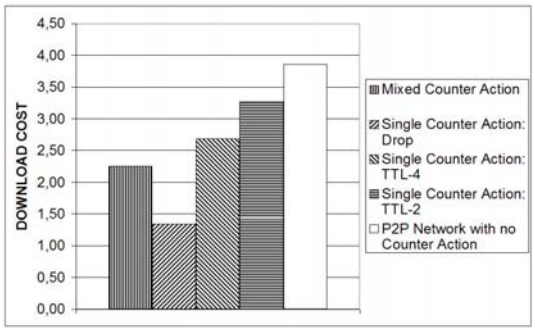

Figure 6.7: Decrease in contributors' download cost when counter-actions are applied.

when the DROP single counter-action is applied. The framework achieves a 41% reduction when a mixed counter-action scheme is applied.

- *Unsuccessful Downloads:* We also looked at the improvement achieved in the number of unsuccessful downloads when the proposed counter-action schemes are used. As Figure 6.8 shows, the DROP single counter-action achieves the best improvement; the number of unsuccessful downloads is reduced by 97%. The mixed counter-action scheme, on the other hand, reduces the number by about 70%. The decrease in the number of unsuccessful downloads means that contributors can better access the network resources when the proposed mechanisms are used. Free riders' requests and downloads may prevent non-free rider peers from accessing files and other resources. When the traffic due to free riders is reduced, the contributors start reaching to the resources more easily and get better satisfied with P2P network services.

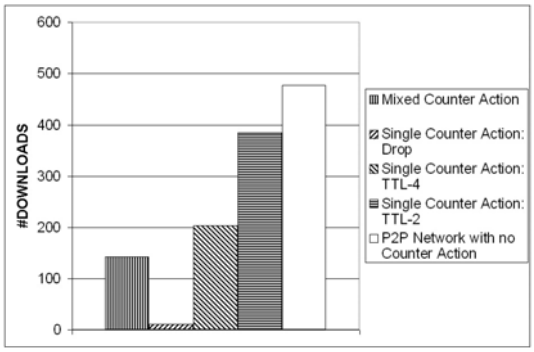

Figure 6.8: Decrease in contributors' unsuccessful downloads when counter-actions are applied.

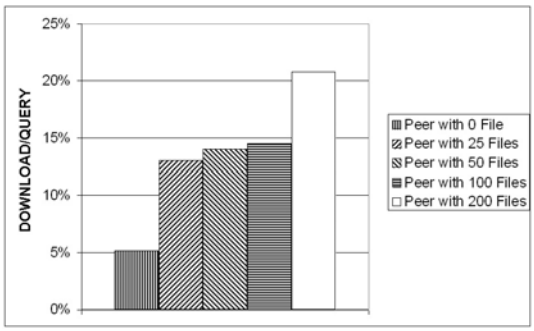

Figure 6.9: Increasing utility values for increasing number of files shared by a probe node.

- *Fairness:* To observe the fairness of DPM, we conducted several simulation experiments. In these experiments, we randomly chose a probe peer and assigned to it different number of files to share. As seen in Figure 6.9, we assigned to the probe peer none (0), 25, 50, 100, and 200 files, and observed the Download/Query ratio (number of downloads / number of submitted queries) as an indication of peer's utility from the system.

  As the figure shows, although the probe peer submits nearly the same number of queries, it can download different number of files depending on how much files it shares. Because, when it shares less files while requesting the same amount of service, it will face counter-actions, and this will limit the number of downloads it will be able to get. On the other hand, when it shares more files, monitoring peers will not apply any counter-action, thus it will be able to reach more peers and download more files. Therefore, if

two peers have similar query patterns but provide different levels of service to the system, they will get different levels of utility from the system as well. Thus, DPM is fair. In other words, a peer contributing more than what is needed to overcome the threshold is fairly compensated. Hence, DPM does not only encourage peers to provide enough services to overcome the threshold barrier, but also encourages them to contribute more to get better service.

6.1.4 Effects of Different Parameter Values

We also executed sensitivity experiments to observe how DPM performs for different values of important parameters: the number of peers and the level of free riding. We also observed the performance results for different values of TTL modifying counter-action. We found that in spite of different parameter settings, DPM provides consistent performance gains.

- *The number of peers in the simulated network:* Considering the size of the Gnutella network, the number of peers simulated in our work can be considered to be very small. However, since our detection and counter-action mechanisms require only local interactions between neighbors, the number of peers in the network should not affect the performance of the proposed approach. This is indeed what we have observed in the results of our experiments that are performed for various network sizes: 400, 900, 1600, and 2500 peers. Figure 6.10 displays the performance results in terms of the number of downloads by free riders. As shown in the figure, the decrease in the number of downloads of free riders is around %50 for all four different network sizes. Therefore, we conclude that increasing the number of peers in the network does not affect the performance of our framework, and our framework is scalable.

- *The Size of Free Rider Population:* We also observed the effect of the size of free rider population in terms of the three metrics mentioned above. As seen in Table 6.5, regardless of the ratio of free riders, our framework achieves a reduction in the number of downloads of free riders. For a smaller ratio

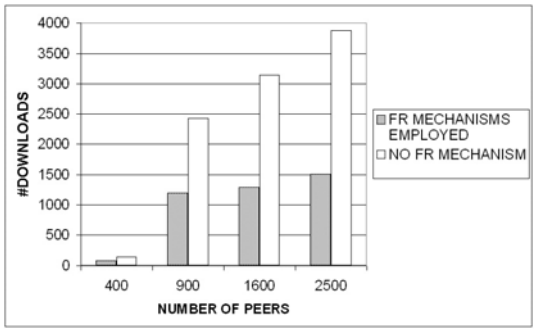

Figure 6.10: Decrease in free riders' downloads when different numbers of peers are simulated.

FR Population (%)	# Downloads with DPM	# Downloads without DPM	Change(%)
60%	554	1306	-58%
70%	562	1142	-51%
80%	631	1219	-48%
90%	544	901	-40%

Table 6.5: Effect of free rider population on the number of free riders' downloads.

of free riders in the overall population of peers, the reduction in downloads of free riders is more pronounced (50%). For a higher ratio of free riders, however, the reduction is still good and is about 40%.

For the second metric, the number of downloads of contributors, the results show that as the size of free rider population increases, our framework provides more downloads for contributors. As seen in Table 6.6, the increase in the number of downloads by contributors reaches up to 36%.

Table 6.7 shows the impact of free rider population ratio on the messaging

FR Population (%)	# Downloads with DPM	# Downloads without DPM	Change(%)
60%	711	701	1%
70%	438	391	12%
80%	314	266	18%
90%	105	77	36%

Table 6.6: Effect of free rider population on the number of contributors' downloads.

FR Population (%)	# P2P messages with DPM	# P2P messages without DPM	Change(%)
60%	973189	1994822	-51%
70%	826705	1932032	-57%
80%	768145	1706333	-55%
90%	711429	1736857	-59%

Table 6.7: Effect of free rider population on the number of P2P messages of all peers.

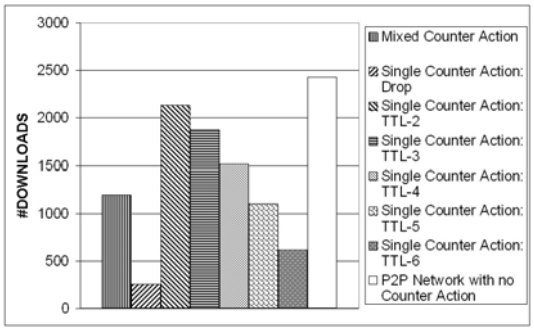

Figure 6.11: Downloads of Free Riders when different counter actions are employed.

overhead in the network. As the ratio of free riders increases, the gain that we achieve with our framework also increases. When, for example, the ratio of free riders is as high as 90%, the reduction in P2P control traffic seen in the network as a result of the application of our framework is 59%.

- *Modifying TTL with different values:* We would like to provide some of the results we obtained when different values were used to decrease TTL other than the default value 1. In the figures 6.11 and 6.12, it can be observed the performance effect of TTL-2, TTL-3, TTL-4, TTL-5, and TTL-6 along with Mixed and Drop actions. As seen in Figure 6.11, TTL-2 has the least effect while TTL-6 is most effective in reducing the download of free riders. We also provide the results in terms of the reduction in the number of P2P messages of free riders in Figure 6.12.

  The level of the effect of modifying TTL counter action is increasing with the decrement value applied. That is, if we use a large decrement value, e.g. 5 or 6, the positive effect of the counter action increases. As expected,

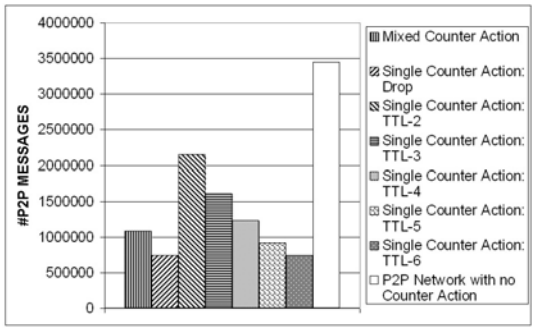

Figure 6.12: The Number of P2P messages of Free Riders when different counter actions are employed.

we observed that TTL-2 had the least improvement on the performance and, TTL-6 and TTL-5 yield similar results to that of the DROP counter action. However, TTL-4 produced a mid point between them. Therefore, to give some insight of the effect of modifying TTL action on the system performance, we select TTL-2 and TTL-4 as representative values.

6.1.5 Possible Attacks

In this section, we describe a list of possible counter attacks against DPM. We also discuss how our framework would react and how we can defend against those kinds of attacks.

6.1.5.1 Fake QueryHit Messages

A free rider can cheat its neighbors (monitoring peers) by replying to some queries with `QueryHit` messages fraudulently as if it has the requested file. When the requesting peer asks for the file, it may just refuse uploading it. In this way it may pretend as it is serving well, since controlled peers may not be aware of unsuccessful download and cheating. In the log tables of its neighbors, the malicious peer may seem to be a non-free rider because of its `QueryHit` replies.

Given the descriptors in Gnutella protocol [18], it may not be possible for a controlled peer to observe and perceive this kind of fake messages. Because, download occurs between two peers outside the P2P network and there is no

Descriptor	Description	Content
`Notify`	Used to report a suspected peer that refused to upload the file it provided in `QueryHit` descriptor in respond to a given `Query` descriptor.	`Query` Descriptor Id; Suspected peer IP; File Index

Table 6.8: New Protocol Descriptor

feedback mechanism for downloads in unstructured P2P networks. To handle this kind of fake `QueryHit` messages, we propose to use a new descriptor: `Notify` (see Table 6.8). This descriptor is used to report about a malicious peer to its neighbor. When a querying peer is refused by a responding malicious peer during a download attempt, the querying peer may send a `Notify` descriptor through the P2P network to reach the monitoring neighbor of the malicious peer. To avoid an increase in the network traffic, the querying peer does not broadcast the descriptor message. Instead, it forwards the descriptor only to the neighbor which has delivered the `QueryHit` message, containing the IP address of the denying peer. Any intermediate peer on the way to the denying peer forwards the `Notify` message to only one of its neighbors based on the message ID (GUID) of the `Query` message stored in its query routing table[3]. The monitoring peer on the path to the denying peer is the neighbor of the denying peer. After processing the `Notify` message, the last peer logs this message, and takes the necessary action against the malicious peer.

There could be some side effects of the proposed `Notify` descriptor. A malicious peer can initiate an application-layer Denial of Service (DoS) attack using the Notify messages. However, almost every message type in P2P protocol (Query, QueryHit, Push, Ping, and Pong) can be exploited in order to launch denial of service attacks [21, 23, 24, 63]. Some proposals exist in the literature aiming to counter the application-layer DoS attacks [23, 76, 86]. We think that we can also use some schemes to deal with DoS attacks using the Notify message. One scheme can be based on the comparison of the number of Notify messages routed

[3]Since, as a requirement of Gnutella P2P Protocol, the `Query` messages are stored in the routing table of each peer for some time to route back the possible `QueryHit` messages, we do not need to store extra state information that can be used to route the `Notify` message on intermediate peers.

by each controlled peer. If a monitoring peer detects a big difference among the number of Notify messages routed by its controlled peers, it can begin to filter (delete/drop) Notify messages coming from that controlled peers (similar to what is proposed in [23]). Since the danger of DoS attack exists for all P2P protocol messages, we think that the precautions taken for other P2P messages can be applied for Notify message as well. Prevention of DoS attacks is out of the scope of our current work, however, it can be interesting to investigate the applicability and effectiveness of the two simple schemes described above as a future work.

6.1.5.2 Fake Files

Free riders could also share dummy files with popular names in order to cheat querying peers. These files can be very small in size to reduce upload overhead. In that way, free rider peers can conceal themselves. This situation however, can also be prevented by using the `Notify` descriptor proposed above.

6.1.5.3 Hiding Query Ownership

In our free riding detection mechanism, the monitoring peers exploit the TTL field value of the incoming `Query` messages to decide if the controlled peer is the owner of the message or not. If the TTL value of the `Query` message is equal to the max TTL value, then the `Query` message is assumed to be originated at the neighbor.

If a free rider wants to prevent monitoring peers applying the counter-actions against its queries, it may try to hide its ownership of the queries by setting the TTL field to a value different than the standard maximum TTL value. Then the originator of the `Query` will not be identified correctly by a monitoring peer. If the free rider sets TTL to a value greater than the allowed maximum value, this can easily be detected by the monitoring peer. If the free rider sets TTL to a value less than the allowed maximum, then the free rider harms itself by reducing the search horizon of the `Query`. In this case we think that there is no need to take an extra action, since we expect that a free rider will not decrease its search

Metric	Standard TTL	Malicious TTL	Change (%)
# Downloads of FRs	1198	840	-29.90%
# Downloads of non-FRs	920	937	1.85%
# P2P Messages of FRs	1086650	842450	-22.47%
# P2P Messages of all peers	1973761	1675965	-15.09%
# Uploads of non-FRs	2094	1757	-16.09%
# Unsuccessful Downloads of non-FRs	147	62	-56.34%

Table 6.9: Results of free rider (FR) malicious TTL attack (mixed counter-action applied).

horizon voluntarily[4].

We observed the effects of this kind of malicious action in our simulations, and Table 6.9 provides the results[5]. During the experiments, we assumed that all free riders act maliciously with regard to the initial TTL value setting in `Query` messages. This is the worst case for our framework. We argue that although free riders may prevent the monitoring peers from applying counter-actions by using malicious TTL values, their level of benefits from the system and their negative effects on the system will also decrease considerably if they set the TTL value maliciously. When they cheat on TTL, they actually reduce the reach of their own queries, and hence the quality of the results they get. As Table 6.9 shows, when a malicious TTL value is used, the amount of downloads of free riders decreases. The number of P2P messages observed in the network due to free riders also decreases. Hence, acting maliciously on the TTL value does not help to the free riders. Therefore, we do not see an urgent need to develop a solution against this kind of TTL attack.

[4]This is because using an initial TTL value even one less than the allowed maximum decreases the search horizon dramatically. For example, if a free rider submits a `Query` message with an initial TTL value of 6 in a network where the maximum allowed value is 7, then the free rider loses about 67% of its reach (search horizon) compared to submitting the `Query` with a TTL value of 7.

[5]We have used the mixed counter scheme while performing simulation experiments for evaluating the effects of attacks to the framework.

Metric	No Peers apply DPM	Only Contributors apply DPM	Change (%)
# Downloads of FRs	2430	2130	-12.30%
# Downloads of non-FRs	840	853	1.5%
# P2P Messages of FRs	3449416	2672604	-22.51%
# P2P Messages of all peers	4679878	3791657	-19%
# Uploads of non-FRs	3216	2939	-8.60%
# Unsuccessful Downloads of non-FRs	477	413	-13.40%

Table 6.10: Results of free riders (FR) insufficient cooperation attack (mixed counter-action applied).

6.1.5.4 Insufficient Cooperation Against Free Riding

Some peers may be reluctant to use the proposed mechanisms against free riding or malicious peers may collude with their neighbors to hide each other's "free-riding status". Thus, free riders may attack the system by disabling the proposed framework. As a result, we may observe low level of cooperation against free riding due to the high population of free riders. We have simulated such an environment by applying the worst-case scenario (all free riders collude) and observed the results. We have compared the case when our framework is applied by only contributors with the case when our framework is not applied at all. In Table 6.10, we provide the results for both cases.

As Table 6.10 shows, even though only 30% of peers apply the mechanisms (they are contributors), the number of downloads of free riders is decreased, the messaging overhead is reduced, and the load on contributors is decreased compared to the case when our framework is not applied. This implies that our mechanisms are quite robust against the type of attack where some peers disable the proposed mechanisms by either collusion or modifying their client software.

6.1.5.5 Constantly Changing Neighbors

A free riding peer may attack the framework by constantly changing its neighbors, and thus it may keep utilizing the services without ever being identified as a free rider.

As discussed in Section 3.1, the P2P network traffic observations [84, 92, 39] show that peers tend to stay connected quite long periods of time. One of the reasons for that is the practical difficulty of disconnecting and re-connecting again. Another reason is that a peer does not get query hit messages immediately after it has submitted a query. A peer should not change its neighbors for the time period between submission of a query and the arrival of the respective query hits (name it *search-QueryHit cycle duration*). If the peer breaks the existing links too fast, it will not get a reply. Therefore, the peer should stay connected for at least a certain time interval which should be longer than the search-QueryHit cycle duration.

Hence, if our scheme can detect a free rider and apply a counter-action against it in a time interval that is less than the search-QueryHit cycle duration, then the attack will not work and it will not make much sense for a free rider to try this. Therefore it is important to know how long it takes to get query hits back and how long it takes to detect the free riders. These concerns depend on several factors. The success ratio (the ratio of free riders that are detected correctly) can give us a clue about the speed of our detection mechanism. Figure 6.13 plots the success ratio versus simulation time. At the beginning of a simulation run, the success ratio will be zero since there is no free rider detected yet. Towards the end of the simulation run, however, the success ratio will have a value that can be close to 1 in ideal case.

In Figure 6.13, we observe that, with the default settings of simulation parameters, at time 90, 40% of free riders are detected successfully. At time 150, 60% of free riders are detected successfully[6]. From the figure we can see that free riders start becoming detected after 50 time units. Therefore, if a free rider peer would like to avoid detection, it should change its neighbors every 50 time units, with the default parameter settings. If it changes its neighbors at a rate slower than this, let's say every 100 time units, the chance to be detected and to face counter-actions becomes increased. The probability of detection becomes around 45% for 100 time units.

[6]If the P2P network traffic becomes higher (i.e., more queries are forwarded), the time required to exceed the τ_{QT} threshold will be sooner and free riders will be detected faster.

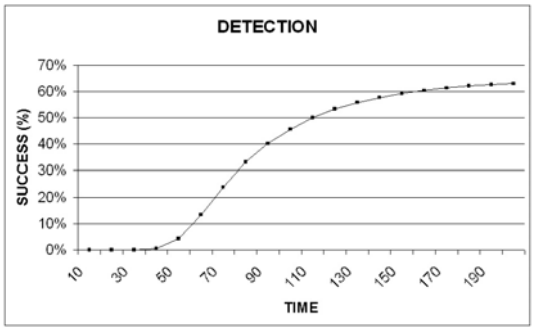

Figure 6.13: The Success of the detection mechanism in the first 200 simulation time.

To investigate the effectiveness of the potential attack, we modified our simulation code to simulate this attack, and conducted several sets of new experiments. In these experiments, we first randomly selected a probe peer to act as a free rider applying the attack. During a simulation run, the probe peer changes its neighbors periodically using a fixed time period between changes. We measured the utility the probe peer gets from the P2P network at the end of a simulation run. The utility is expressed as the ratio of the number of downloads the probe peer performs to the number of queries it submits. We obtained results for two different time intervals between changes of neighbors: 50 and 100 time units. The results are displayed in Figure 6.14. In the figure, we also included two other utility values. One is the utility value that a contributor peer can get and the other is the utility value that a free rider who is not trying the attack (i.e., not changing connections) can get.

As can be seen in Figure 6.14, the probe peer succeeded to increase its utility by changing its neighbors constantly. We can observe that the length of the time period between changes has an effect on the service the probe peer receives, as we have discussed above. If this period is longer, the probability of detection gets increased and the probe peer will more likely face counter-actions; and this will reduce the service it will get.

However, the first experiment we describe above can not reflect a real-life scenario where lots of peers would like to apply the attack at the same time. Therefore we also conducted experiments for the scenario where all free riders in the network apply the attack expecting to increase the utility they get. The results

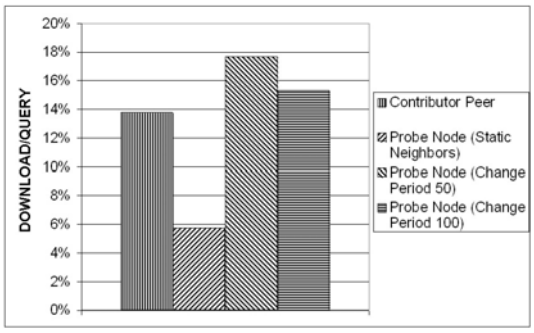

Figure 6.14: The results for the Probe peer, when the attack is only applied by the probe free riding peer.

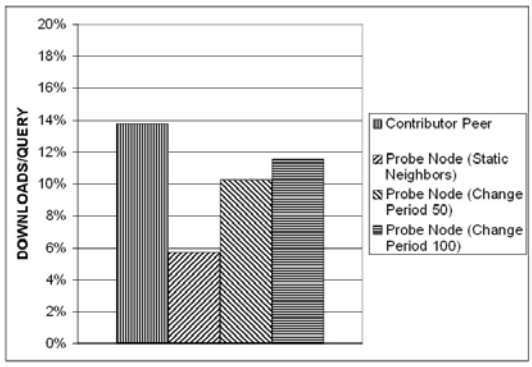

Figure 6.15: The results for the Probe peer, when the attack is applied by all the free riding peers.

are displayed in Figure 6.15. As seen in the figure, the probe node acting as a free rider and applying the attack is negatively affected in this case when all free riders in the network apply the attack. This is because, one of the side effects of the suggested attack is that when all free rider peers change their neighbors, their previous neighbors lose the connection via these peers and they would lose the possible incoming `QueryHit` messages as well. Since the `QueryHit` messages in unstructured P2P networks are routed back through the same route of the received `Query` messages, when an intermediate peer tries to route a `QueryHit` which is routed by a free rider peer, it could not route it anymore, due to the changed neighbors. So, some of the `QueryHit` messages would be dropped without reaching to the destined peers. As observed in the figure, this side effect is not negligible. The probe peer loses its advantage considerably when all other free riders also apply the same attack.

Therefore, we can conclude that although the attack seems to increase the utility of an individual free rider, in a more general and real situation, when all or most of the free riders apply the attack, the utility that a free rider gets is not increased to a level to justify the practical difficulties of applying the attack. The free rider will not reach to a level of utility comparable to that of a contributor peer.

6.1.5.6 Increasing Number of Neighbors

A free rider can try to enlarge its search horizon by increasing the number of neighbors it connects to. It is not easy to totally prevent them doing so without changing the nature of unstructured P2P systems and without loosing major advantages of these systems. However we believe that the attack is not so practical for free riders to apply and it does not lead a significant increase in their utility. Below, we would like to provide a simple analysis for the effectiveness of the attack. Later, we share some results of the simulation experiments obtained when the attack is applied.

Assume that in a P2P system, average number of connections per peer is 4 and maximum TTL is 7. If a free rider employs the attack as suggested above, it can connect to new nodes and but soon after it would be detected by these peers as well and be subjected to some counter-action. Therefore, its messages' TTL will be decreased always to some value or will totally be dropped. If TTL-4 is implemented as a counter-action, the decremented TTL of the free rider's Queries would be 3 (7-4). Now, we would like to find the number of peers to be connected to provide the same amount of connectivity when TTL is 7. As a general assumption, we think that the probability of getting QueryHit messages to Queries is positively correlated with the number of peers connected. Therefore, the attack suggests connecting more peers even with reduced search horizon. When TTL is 7, a peer can connect to 4372 peers at most (4 connection per peer is assumed) When TTL is 3, the peer can reach 52 peers at most. Therefore, it loose 4372-52=4320 peers. To compensate this, it will try to connect to other peers with TTL 3 (because, as its new neighbors will discover it as a free rider sooner). Therefore, each newly connected neighbor can provide at most 17 peers (including itself). To have the same amount of connectivity, free riding peer

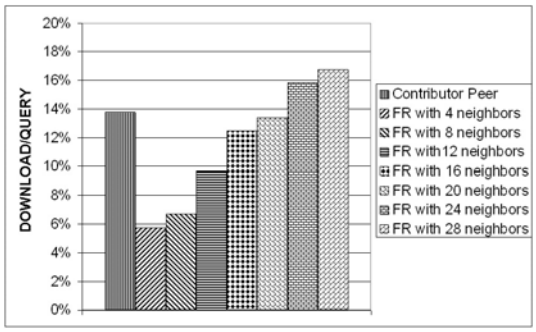

Figure 6.16: The results for the Download/Query ratio, when the increased number of neighbors attack is applied by a probe peer.

should connect to (4320/17)= 254 new peers. That is, while average/contributor peers have 4-peer connectivity to cover the same number of peers, free riding peer will have to make about 64 times more connections (total 258 connections). We think that it is not easy and practical to do. One of the practical drawbacks is the fact that more neighbors mean more P2P messages to process, which could create a big burden on the peer considering the high amount of P2P traffic in real life application. Even though the peer may choose to drop these messages, they will still reach to the application layer and will degrade the performance of the peer's system.

We have conducted several experiments to observe the effects of the suggested attack. As seen in Figure 6.16, download/query ratio for a contributor peer with 4 neighbors is about 14%. On the other hand, a free rider peer with the same amount of connection has download/query ratio of only 6% due to the Mixed counter-action applied. It is an expected result and in line with the prior results provided. Even we increase the number of connected neighbors five times, free rider peer could not attain the same download/query ratio of a contributor peer. When the free rider peer has 6 or 7 times more connections, it then exceeds the download/query ratio of a contributor peer.

Another important observation from the experiments is that the increase in the number of neighbors comes with a cost, i.e. the increasing number of P2P messages. The free rider peer with more connections should devote much more resources to process P2P messages. For example, a free rider peer with 6 times more connections has to deal with almost 8 times more P2P messages(see Figure 6.17).

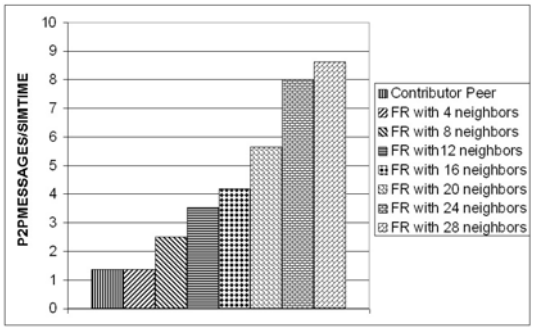

Figure 6.17: The results for the P2P Message/Simulation time ratio , when the increased number of neighbors attack is applied by a probe peer.

We believe that in a much larger P2P network, this could easily be a bottleneck for the peer. The message queue could easily be overloaded and overflowed.

As a result, we could state that there can be some ways to attack, but free riders will experience various difficulties in applying them, due to our proposed mechanisms.

6.2 Simulation Results for the P2P Connection Management Protocol (PCMP)

Below we first present the assumptions and parameter values involved in the simulation experiments. Then, we explain the performance metrics used in evaluating the proposed framework. Later, we provide and discuss the results observed in terms of the performance metrics. We also investigate the effect of different parameter values on the results. As a last discussion, we provide some possible attacks to the proposed framework and their effects on the system.

6.2.1 Assumptions

Assumptions made for the simulation experiments are similar to those presented in Section 6.1.1 and used in the evaluation of DPM. We would like to remind the important parameter settings used before. Our model simulated a P2P network

Property	Contributors	Free Riders
Population ratios	30%	70%
Ratio of shared files of each peer type to total files	99%	1%
Peers replicate the files they have downloaded	True	False
Mean time between queries (exponentially distributed)	60 time units	60 time units
Maximum simultaneous uploads	10	10

Table 6.11: Properties of peer types.

of 900 peer nodes. The peers were inter-connected to form a mesh topology at the beginning of a simulation run. For the base experiments with only the Gnutella protocol (i.e. without PCMP), we assumed that all the peers stayed connected in the same way until the end of simulation run.

We assumed that there were two types of peers in the simulated network: contributors and free riders. The properties of each peer type are summarized in Table 6.11.

There were 9000 distinct files, with four copies of each, distributed to the peer nodes at the beginning of each simulation run. These 36000 files were distributed among the peers and shared according to the file sharing ratios shown in Table 6.11. For the base experiments, we assumed that each file was of the same size and could be downloaded in 60 units of simulation time. In Section 6.2.4 we relax this assumption.

During a simulation run, peers randomly selected files to search for download, and they submitted search queries for them. The inter-arrival time between search requests generated by a peer followed an exponential distribution with a mean of 60 time units.

Each peer's upload capacity (the number of simultaneous uploads the peer could perform) was limited to 10. If a peer reached its upload capacity, any new upload request was rejected. The querying peer could then try to download the file from another peer, selected from a list obtained from the Query Hit message. We assumed that the querying peer would repeat the same request a maximum of

three times. After that, the peer would give up and could initiate a new search for another file. We assumed that TTL is set to be 3 hops. Simulation experiments are run for 4000 units of simulated time, repeated 10 times, and plotted on a 95% confidence interval.

In order to match the topology of the base model, we assumed that each peer could provide up to four IN- and four OUT-connections. This is because the base model compared with PCMP has a mesh topology with an average of four connections per peer.

6.2.2 Performance Metrics

To evaluate our protocol, we defined and studied two families of metrics: 1) topology-related metrics, 2) performance-related metrics. Using the first type of metrics, we aimed to investigate the change in the P2P network topology in favor of contributing peers. The details of the topology-related metrics are presented below.

- *Total number of connections among contributors:* We count the number of connections (IN and OUT) which connect the contributors directly to each other. We expect that if the number of connections among contributors is increased, the contributors will get better service from the network. Since we assume the number of connections that a peer can have to be limited, those connections have to be used carefully by contributors. In order to get better service and more Query Hits, a contributor should have more connections to other contributors and less connections to free riders. In this way, a contributor can also reduce free riding through itself. This metric also shows how successful the PCMP protocol is in discovering and connecting contributors.

- *Total number of OUT-connections from free riders to contributors:* As stated in Section 4.2, if a peer has an OUT-connection to another peer, the peer can submit queries through this connection to that peer. Hence, the number of OUT-connections a peer has increases its chance to get replies and service from the network. Therefore, we count the total number of

OUT-connections that free riders have towards contributors to measure how effective our protocol is in reducing free riders' access to resources.

- *Number of isolated free riders:* One of the aims of our protocol is to isolate free riders from contributors in the P2P network. If a free rider has no OUT-connection, then it cannot send any query and cannot receive any service, and we consider such a peer to be isolated. An isolated peer cannot download any files from the network. The greater the number of isolated free riders, the better it is for the network.

The second type of metrics that we defined are related to the performance and service the peers get from the network. They are used to measure the performance and service improvement in the network when PCMP is employed.

- *Number of downloaded files*: This is an important metric indicating the number of downloads that can be performed in a P2P network during a fixed time interval. If peers can download more files from the P2P network, then the level of satisfaction with the network will be higher.

- *Number of uploads by contributors*: This metric indicates the load imposed on a peer. Contributors can become overloaded due to the excessive number of search and download operations executed mainly by free riders.

- *Download cost*: We define the download cost for a peer as the ratio of the number of uploads to the number of downloads performed by the peer. This ratio indicates the load imposed on a peer compared to the service the peer gets from the network. The smaller this ratio is, the better it is from the perspective of the peer.

- *Number of P2P network protocol messages*: This metric is an indication of the messaging overhead in the P2P network and the underlying infrastructure. Messaging overhead affects the scalability of a P2P system. The messaging overhead may be high due to the flooding approach used in querying, particularly in unstructured P2P networks. A high number of protocol messages sent over the network also increases the level of congestion in the network.

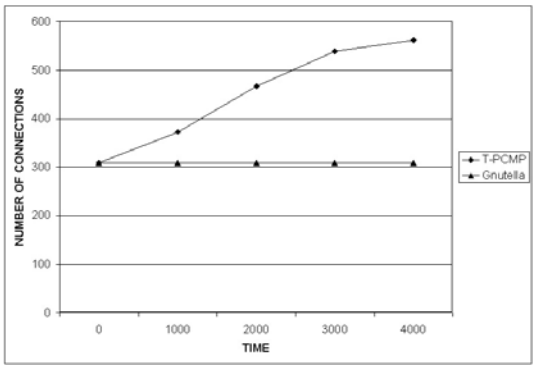

Figure 6.18: Increase in the number of connections among contributing peers.

- *Fairness*: Fairness metric shows that the level of service that can be obtained by a peer is proportional to the level of contribution that is provided by that peer. In other words, a peer contributing more is fairly compensated with more services.

6.2.3 Simulation Results and Analysis

In simulation experiments, we first tested the effectiveness of PCMP in connecting the contributors to each other. Afterwards, we conducted experiments to observe changes in the performance when PCMP is employed.

6.2.3.1 Impact of PCMP on Network Topology

Figure 6.18 shows the number of connections established among contributing peers over the simulation time. The results are for a P2P network employing our PCMP protocol using the time-based replacement policy (T-PCMP). As seen in the figure, the protocol causes more contributing peers to become directly connected to each other as time passes. By the end of the simulation, the number of connections (IN and OUT) among contributors increased from 309 to 562. Hence, connectivity among contributors increased by 82%.

Figure 6.19 shows the number of OUT-connections of free riders to contributing peers plotted against the simulation time. As can be seen in the figure, the protocol caused the number of OUT-connections of free riders to decrease by

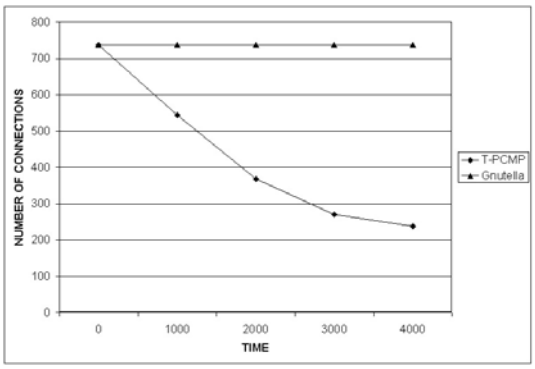

Figure 6.19: Decrease in the number of OUT-connections from free riders to contributors.

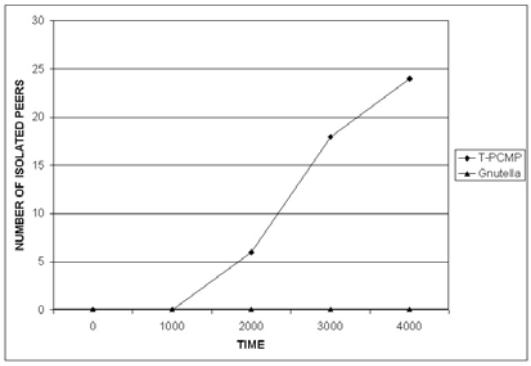

Figure 6.20: The number of isolated free riders.

about 67% by the end of the simulation. This is because when contributors cannot download from free riders over time, they start dropping their IN-connections from free riders; hence free riders lose their OUT-connections to contributors.

Figure 6.20 shows the number of isolated free riders over time. As time has passed, more free riders were isolated from the network (they lost all their OUT-connections). At the end of the simulation time, a total of 24 free riders (out of 630) were isolated.

These results show that PCMP updates the topology effectively according to the contributions of peers: it increases the connectivity among contributors, reduces the connectivity of free riders towards the contributors, and can totally isolate some free riders from the P2P network.

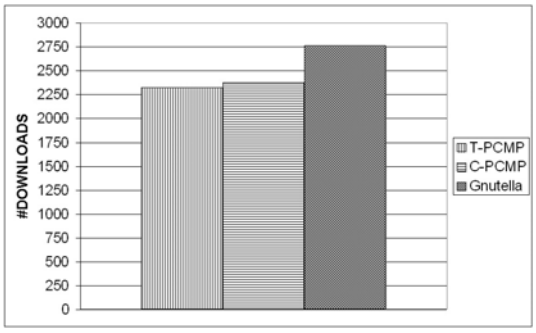

Figure 6.21: Decrease in free riding peers' downloads.

6.2.3.2 Impact of PCMP on P2P Network Performance

This section evaluates the effectiveness of our protocol in terms of the performance metrics described in Section 6.2.2.

Downloads of free riders: As Figure 6.21 depicts, the number of downloads by free riders dropped when PCMP was applied. PCMP decreases OUT-connections of free riders towards contributors, and this reduces the chance of getting a hit on the queries. In this way, the number of downloads by free riders is reduced. Both C-PCMP and T-PCMP reduces the downloads. C-PCMP caused a 14% reduction, whereas T-PCMP achieved a 16% reduction.

Downloads of contributors: It is desirable to increase the number of downloads for contributors. Since each peer's upload capacity is limited, the download requests of contributors can sometimes be rejected. The rate of rejection is higher when there are many free riders in the system, so eliminating the effects of free riders on the P2P network will help to increase the number of downloads that contributors can make. This is indeed shown in Figure 6.22; applying our PCMP methods achieved an increase in downloads performed by contributors by 51%.

Figure 6.22 shows that the improvement in downloads is slightly higher with T-PCMP than with C-PCMP. While T-PCMP yielded an improvement of about 51%, the improvement when C-PCMP was used was about 46%.

Uploads of contributors: A metric that can indicate the load on a peer is the number of uploads performed by the peer in a given time period. As shown above, PCPM increments the number of contributors' downloads considerably

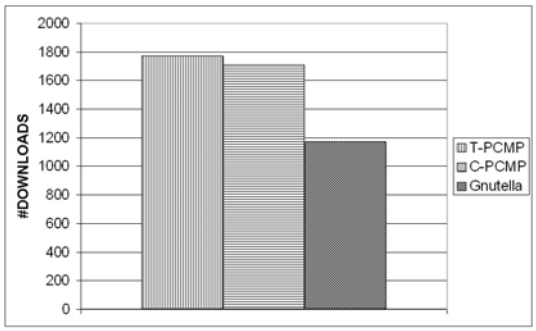

Figure 6.22: Increase in contributors' downloads.

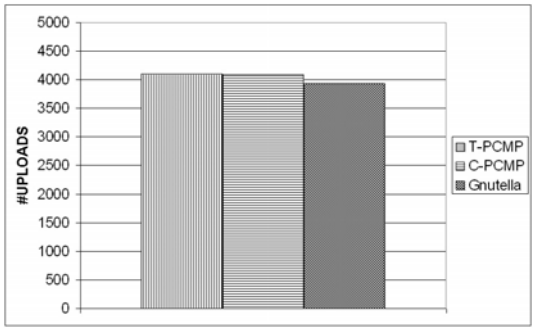

Figure 6.23: Change in contributors' uploads when PCMP is applied.

and reduces the downloads of free riders significantly. In simulation experiments, we observed a slight increase (about 4%) in the number of uploads performed by contributors when PCMP is applied (see Figure 6.23). However, this slight increase is the result of PCMP's success in reconnecting contributing peers to each other. Contributors can download more from each other due to the fact that their searches can reach more contributors by applying PCMP.

Download Cost: The load on a contributor can also be defined as the ratio of its uploads to its downloads. The results of our experiments show that PCMP also causes a reduction in the download cost of contributors. As shown in Figure 6.24, both T-PCMP and C-PCMP achieve a reduction of about 30% in the download cost for contributors.

Number of P2P protocol messages: The number of P2P protocol messages transmitted in the network is an important factor affecting scalability and bandwidth efficiency. PCMP results in a reduction of up to 36% in the number of transmitted P2P protocol messages (Query and Query Hit messages) originating from and

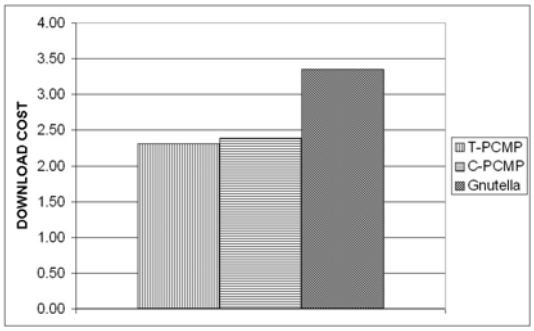

Figure 6.24: Decrease in contributors' download cost.

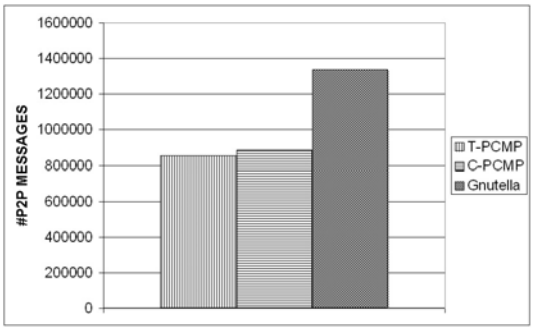

Figure 6.25: Decrease in P2P messages from free riders.

destined to free riders (Figure 6.25). This result shows that applying the proposed PCMP helps a P2P network to handle more peers with less P2P messaging overhead and the system becomes more scalable with respect to the peer population. The reduction observed in the number of protocol messages is the result of reducing or stopping the propagation of Query messages from free riders. As the number of OUT-connections of free riders gets reduced, the propagation of Query and Query Hit messages for free riders will get reduced as well. The reduction of control traffic in a P2P network also means a reduction in the overhead imposed on the underlying infrastructure. This reduction translates to a better utilization of available bandwidths and to a decreased processing load on each peer.

Fairness: We explored how PCMP reacts to the changes in the behavior of peers. A peer can behave as a free rider at first, but later, after observing the decrease in the service it gets, begin to share its resources. If PCMP does not react to these kinds of changes, it will be unfair and moreover it cannot accomplish one of its primary goals, promoting contribution.

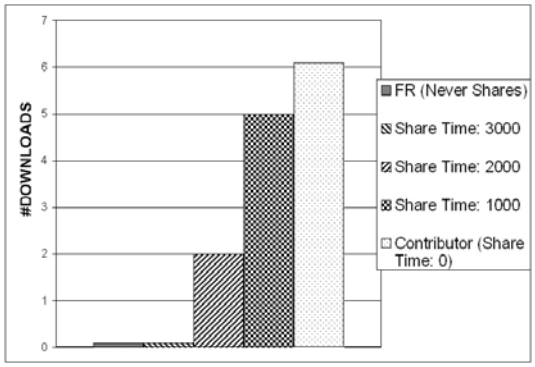

Figure 6.26: Downloads of the probe node according to when it begins to share its files.

To observe the fairness of PCMP, we conducted the following experiment. We randomly selected a probe node which initially behaved as a free rider. After a certain amount of time, the node changed its sharing attitude and began to share its files. We compared the level of service it got from the P2P network when it was behaving as a free rider and when it was sharing its files. The number of downloads that could be performed by the probe peer is depicted in Figure 6.26. As can be seen in the figure, when the peer begins to change its sharing attitude at a given time from free riding to contributing, PCMP reacts in a positive way and allows the peer to download more files.

6.2.4 Effects of Different Parameter Values

We executed sensitivity experiments to observe how PCMP performs under varying values of important parameters: the number of peers and the level of free riding. We also observed the performance results for different file sizes and different levels of file replications. Under all different parameter settings, PCMP was observed to provide performance gains.

- *The number of peers in the simulated network:* Considering the size of a real Gnutella network, the number of peers simulated in our work can be considered to be very small. However, since our proposed method requires only local interactions between neighbors, we do not expect the impact of the number of peers on the network's performance to be considerable.

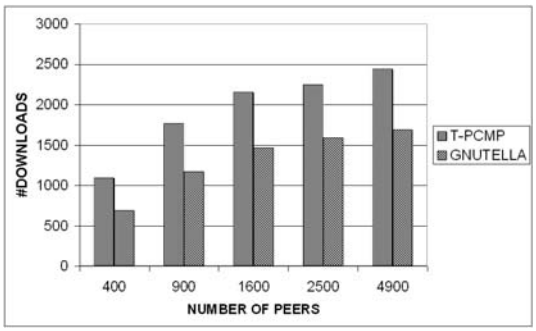

Figure 6.27: The number of contributors' downloads when different numbers of peers are simulated.

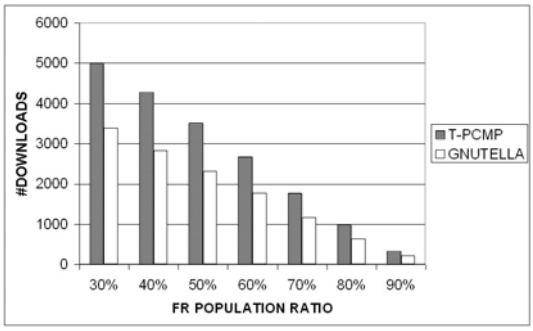

Figure 6.28: The number of contributors' downloads when different free rider populations are simulated.

This is indeed what we have observed in the results of our experiments that were performed for various network sizes: 400, 900, 1600, 2500, and 4900 peers. Figure 6.27 displays the performance in terms of the number of contributor downloads. As shown in the figure, the number of downloads by contributors is increased around 45% for all network sizes. Therefore, we conclude that increasing the number of peers in the network does not negatively affect the performance of our framework, and that our framework is scalable.

- *The Size of Free Rider Population:* We also evaluated the effect of the size of the free rider population. As seen in Figure 6.28, regardless of the ratio of free riders, T-PCMP achieves more downloads, around 50%, for contributors. Even at a low population ratio of free riders, the protocol performs very well.

File Type	File Size	Ratio
Very Small	$\sim$ 0.3	10%
Small	$\sim$5MB	50%
Medium	$\sim$40MB	20%
Large	$\sim$100MB	10%
Very Large	>100MB	10%

Table 6.12: Properties of different file sizes.

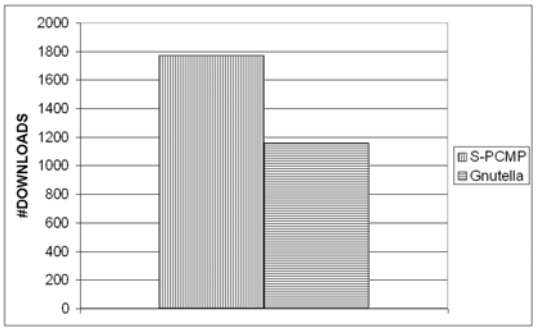

Figure 6.29: The number of contributors' downloads with the existence of different file sizes.

- *The File Size and File Replication:* In Section 6.2.1, we assumed that each file is of the same size and the number of copies for each file is identical. In this section, we relax these assumptions by considering different file sizes as summarized in Table 6.12, and different levels of file replication as shown in Table 6.13. The values given in tables are based on the results of the P2P network observations provided in [64, 65].

  We proposed two connection replacement policies in Section 4.4, namely T-PCMP and C-PCMP. To handle different file sizes, we now propose a new replacement method. In this method, the size of the file downloaded from the neighboring peer is used to select the connection for replacement. The connection with the least total amount of downloaded file is selected as a victim. We call the PCMP protocol that applies this policy *Size-based PCMP (S-PCMP).*

  Figure 6.29 shows the results of different file sizes on the contributor downloads. PCMP increases the contributor downloads as much as 55% compared to Gnutella.

NAME	Group A (Ratio/Replication)	Group B (Ratio/Replication)
RARE	10% of files: 1 copy.	90% of files: 4 copies.
POPULAR	10% of files: 40 copies.	90% of files: 4 copies.
UNIFORM	All files : 4 copies	All files : 4 copies

Table 6.13: Properties of different levels of file replication.

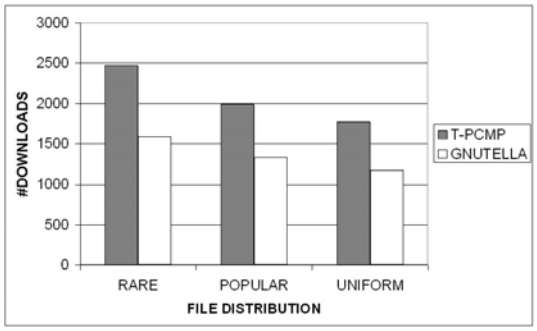

Figure 6.30: The number of contributors' downloads with different levels of file replication.

For evaluating the impact of different file replication levels, we used three file replication schemes as summarized in Table 6.13 . We split the files into two groups and replicated them with different factors. In the RARE distribution, 10% of the files are rare (fewer replications) compared to 90% of the files. Similarly, in the POPULAR distribution, 10% of the files are more popular (more replications) than the rest of the files. In UNIFORM (default) distribution all the files have the same number of copies.

The results of the simulation tests are depicted in Figure 6.30. The figure summarizes the effects of different file distribution schemes on the contributors downloads. With all the file distributions considered, PCMP performs about 55% better than Gnutella. Total number of downloads of contributors is affected by the distribution strategy of file copies. However, PCMP manages to profit the contributors with all different types of file distribution schemes evaluated.

6.2.5 Possible Attacks

There are many different kinds of attacks to the existing P2P network protocols. Since we extend the Gnutella Protocol, we will not discuss the attacks and their effects related to the original Gnutella Protocol. Here we would like to discuss the several possible attacks specific to the method we proposed against free riding.

6.2.5.1 A malicious peer does not comply with the proposed PCMP rules

A malicious peer may refuse to add a contributor to its list of IN-connections after downloading a file from the contributor. We claim that by doing this the malicious peer cannot gain anything. It can only stop incoming Query and Ping messages via its IN-connections. This, however, may decrease the search horizon of the contributors.

If all free riders apply this attack, then contributors establish OUT-connections only with other contributors, and this automatically helps them to become more connected with each other. In the end, contributing peers will have an advantage over free riders, since a peer has a restricted number of OUT-connections and a contributor will not waste them for connections to free riders. Because, as discussed in Section 4.2, if a contributor uploads a file to a peer, the contributor will update its OUT-connection with that peer. If there is no free OUT-connection, then it will drop an existing OUT-connection and add the new peer. If the dropped connection is with a contributor and the newly added connection is with a free rider, the contributor will not benefit from the new connection since free riders do not share files. However, the contributors are not aware if a peer is a free rider or not. If free riders reject IN-connection requests by not sending a Pong message, then the contributors will not update their OUT-connections. The contributors will only update their OUT-connections when they upload files to other contributors, since other contributors will accept the IN-connection requests by replying with Pong messages. Therefore, we expect that this attack will not affect the contributors much.

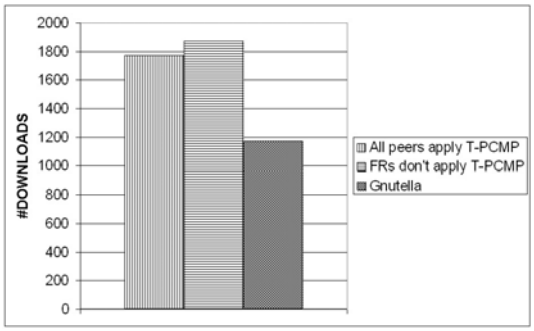

Figure 6.31: The number of contributors' downloads when free riders are noncooperative.

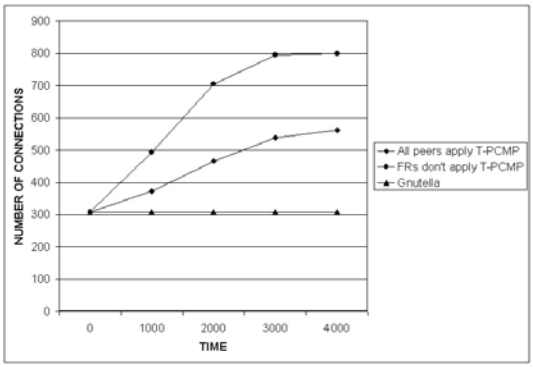

Figure 6.32: Increase in the number of connections among contributing peers when free riders are noncooperative.

In order to observe the effects of this possible attack, we designed a new simulation setting. In this simulation, we assumed that all free riders would reject creating an IN-Connection from a source peer after downloading a file. As seen in Figure 6.31, this attack does not adversely affect the download performance of the contributors as compared to the results given in Figure 6.22. On the contrary, the contributors can download slightly more files, because they become more closely connected to each other, as seen in Figure 6.32.

6.2.5.2 A malicious peer replies with a faked Query Hit

To establish OUT-connections, a malicious peer can reply to a Query message as if it has the file. However, when the querying peer demands the file, the malicious peer can upload a fake file. But this will not help the malicious peer to establish

an OUT-connection. Because, in PCMP, the connection between two peers is established after a file is downloaded, and connection establishment is initiated by the uploading peer through a Ping message. If the downloader peer is not satisfied with the file, it will not send back a Pong message and the connection will not be established. Therefore, the malicious peer cannot use this attack to gain more OUT-connections.

6.2.5.3 A malicious peer behaves as a new-comer to gain more OUT-connections

To increase the number of OUT-connections, a malicious peer can request OUT-connections from peers as if it is a new peer in the network. If the peers accept all newcomers' connection requests without any limitations, the attacker can benefit from this situation. Jakobsson and Juels proposed a method of combating such problems: proof of work (POW) protocols [51]. The main idea of these protocols is that *a prover* demonstrates to *a verifier* that it has expended a certain level of computational effort in a specified interval of time. POWs were proposed as a mechanism for a number of security goals including server access metering, construction of digital time capsules, uncheatable benchmarks, and protection against spamming and other denial of service attacks. In our work, we can implement POW to minimize these attacks to very low levels. Thus creating new connections can cost time, limiting the ability of the attackers to request them without a limit. We can include a rule in the general P2P protocol for initial connections stating that clients are required to solve a puzzle, such as factoring a number, before a Ping request is answered with a Pong message. The puzzles could require additional work as resources become more scarce. This increases the resources required by attackers to attack the system proportional to the threat of the attack. A short lifetime on connections can also help to manage the network and prevent this kind of attack.

6.3 A Discussion on the Comparison of DPM and PCMP

In Sections 6.1.3 and 6.2.3 we presented the detailed performance results of DPM and PCMP, respectively, over a typical pure P2P network. In this section, we provide a discussion about these frameworks comparing their characteristics and performance gains.

6.3.1 Comparing Characteristics of DPM and PCMP

Both frameworks are designed to attack the free riding problem in pure P2P networks by limiting the free riders' ability to exploit the network resources. However, their approaches to locating and countering free riders are considerably different. As a result, their effects on the performance of a generic P2P network differ as well. Below we first remark the similarities and differences between these two frameworks, and then we summarize the simulation results with respect to some important performance metrics, and discuss their advantages and disadvantages over each other.

The similarities of the frameworks are as follows:

- DPM and PCMP frameworks are simple to implement, have low overhead to run, fully comply with the concepts and protocols of pure P2P networks, and are decentralized to operate efficiently.
- Both of the frameworks detect the free riders by monitoring local interactions and mutually provided services.
- Since the frameworks solely depend on interactions between two peers and each peer can make decision about other peers according to local information, the frameworks are scalable with respect to the number of peers in P2P network.
- Neither of them requires any kind of security infrastructures which makes them practical and low-overhead to be implemented in pure P2P networks.

- The frameworks are successful to reduce the impact of free riding on P2P networks with respect to various performance metrics.
- They are resistant against the possible attacks of malicious peers.
- As we designed these solutions as frameworks, one can implement them in a real world scenario by modifying their various parameters accordingly.

The frameworks have the following important differences:

- DPM's counter-actions mainly either limit the search horizon of free riders or stop the free rider searches flooding into the P2P network. However, PCMP breaks the connection with free riders and modify the P2P network topology in such a way that free riders are pushed away from contributors or, even more, free riders can be isolated.
- Implementation of DPM does not introduce any overhead for the P2P network. On the other hand, PCMP requires updates in routing tables due to changes in connections among peers. This does not necessarily mean a significant overhead for the system. Because, frequent disconnections and unreliable links in pure P2P networks naturally require this kind of updates.
- DPM does not require any change in P2P network protocol whereas PCMP does. A DPM implementation involves only modifying the client software of the underlying P2P protocol. However, a PCMP implementation requires modification in both the client software and the P2P network protocols used to implement and manage a new connection type, called One-Way-Request-Connection (OWRC). Nevertheless, PCMP's modifications required for the network protocols are very simple and can be implemented at the application level.

6.3.2 Comparing Performance Results of DPM and PCMP

A summary for the comparative simulation performance results is provided in Table 6.14. In this table, the range of the given results is due to various implementation of the frameworks.

Metric	DPM	PCMP
# Downloads of free riders	-89% .. -12%	-16% .. -14%
# Downloads of contributors	+10% .. +7%	+51% .. +46%
# P2P messages	-78% .. -68%	-36% .. -34%
# Uploads of contributors	-68% .. -35%	+5% .. +4%
Download cost for contributors	-65% .. -41%	-30% .. -28%

Table 6.14: Summary of DPM and PCMP performance results over the simulated pure P2P network.

When we examine the results, we first notice that both frameworks improve the performance of the base P2P network, namely Gnutella. DPM is much more successful in punishing free riders compared to PCMP. The downloads of free riders can be reduced down to 89% when DPM is employed. PCMP has a limited success on reducing free riders downloads. However, PCMP can boost downloads of contributors considerably, up to 51%. For this metric, DPM can not achieve high performance.

Since DPM intervenes free riders at the search phase, it successfully reduces the number of P2P messages. PCMP can push free riders to the outskirts of the network and can isolate them. However, most of the free riders are still connected to the network and can still submit queries. Thus, PCMP can not reduce the number of P2P messages as much as that observed with DPM.

As a consequence of its success in reducing the downloads of free riders, DPM reduces the uploads of contributors considerably. On contrary, as PCMP boosts downloads of contributors significantly, the uploads of contributors do not diminish at the same level as DPM. These results can be observed by comparing download costs of these two frameworks as well.

In summary, we can argue that both frameworks are successful to fulfill the desired

goal which is to reduce the adverse effects of free riding on P2P networks and prevent free riding peers from exploiting the network resources unresponsively in the course of time. Additionally, we observe performance gains for contributors and performance degradation for free riders by reducing the level of free riding and its effects. Thus, P2P networks employing the proposed free riding mechanisms become more robust and scalable.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this dissertation, we propose two different solutions, Detect and Punish Method (DPM) and P2P Connection Management Protocol (PCMP), to counteract free riding in pure P2P networks. In essence, we aim to reduce the amount of free riding and its negative impacts on P2P networks and peers. As the performance results of simulation experiments indicate, the solutions succeed in improving the performance of the P2P networks in terms of the following metrics:

- the quality of services that peers can get from the network,
- the availability of content and services,
- the robustness of the system,
- the load balance on peers, and
- the scalability of the network.

Furthermore, simulation experiments prove that both solutions can cope with possible counter attacks of malicious free riders and most of these attacks can not render the proposed frameworks obsolete.

In our first solution, DPM, we provide a distributed and measurement based framework against free riding. It is a framework consisting of tunable and changeable algorithms. As part of the framework, we first specify possible types of free riding that can be encountered in a P2P network. Then we propose some mechanisms that can be used to detect free riders of the defined types. We also present sample counter actions that can be applied against the peers detected as free riders. It is shown through evaluating DPM that employing the "dropping single counter action" scheme against all kinds of detected free riders results in better improvements for all the performance metrics used except the number of downloads by contributors. We think that this result is due to false detection in determining free riders. As one would like to increase the performance for contributors, the usage of "mixed counter action" scheme is offered. This scheme is shown to be the best counter action increasing the number of contributor downloads, as well.

Our second solution, PCMP, is a novel approach employing a connection management protocol that can act against free riding in pure P2P networks. The main idea of this solution is to adapt dynamically P2P network topology in order to promote contribution in the network. PCMP manages the connections among peers based on the amount of contributions by peers. Towards the solution, we first provide a new connection type, *One-Way-Request Connection* (OWRC), in which the direction and the type of messages are well defined. Then, we provide a connection management protocol which connects peers to each other based on their interactions. As a result, the contributors discover each other and get connected to each other, while free riders loose connections to contributors, or even they can be excluded from the network. In other words, the modified topology supports the contributors to download more files from the other contributors and process less queries from free riders.

As a conclusion, we show in our dissertation that;

- The impact and the level of free riding in P2P networks can be reduced if proper mechanisms are implemented.
- Two different solutions we proposed succeed in decreasing the downloads of free riders while increasing the downloads of contributing peers.

- Both solutions boost the scalability of P2P networks by significantly reducing the P2P network traffic brought out by free riders.
- Free riders can attack the solutions through various methods, however they can not benefit from these attacks. The proposed solutions can still limit the level of free riding in the presence of such attacks.

7.2 Future Work

As a future work, we plan to refine the proposed free riding types in DPM to enable the detection mechanism work better. We also plan to simulate different network topologies to demonstrate the properties of power-law and small-world phenomena. Furthermore, we plan to integrate game theory into the simulated peers so that peers can follow a strategy to maximize their utility from the system. We finally would like to implement the proposed solutions into a Gnutella client and test the solutions in an existing P2P network so that we can observe the effects of DPM and PCMP in real world applications.

Appendix A

Analyzing Effects of PCMP

We here provide a motivational example about how we can improve the performance in terms of some metrics in a P2P network using the P2P Connection Management Protocol (PCMP) that we presented in Chapter 4.

The probability of getting a query hit depends on many factors including the popularity of the requested file, the number of files shared by peers, and the number of contributing peers in the search horizon. If we assume even popularity and even number of shared files by each peer, then the number of contributing peers in the search horizon will be the factor determining the hit probability of a query. Therefore, increasing the number of contributors in the search horizon is important for receiving better service from the P2P network.

In order to calculate the number of contributors that a contributing peer's query can reach, we first make the following assumptions. In a P2P network there are contributors and free riders. A peer is considered as a free rider if it does not share any files at all. On the other hand, a peer is a contributor if it shares any number of files. A Gnutella-like protocol is used for the query dissemination with the time-to-live (TTL) value set to m. Each peer in the network has n one-hop neighbors on the average. The number of peers in the network is so large that the path followed by a flooded query constitutes a tree, not a graph. In other words, a query reaches distinct peers at each hop while getting flooded from one hop to the next. A contributor has p number of contributor neighbors and $n-p$ number

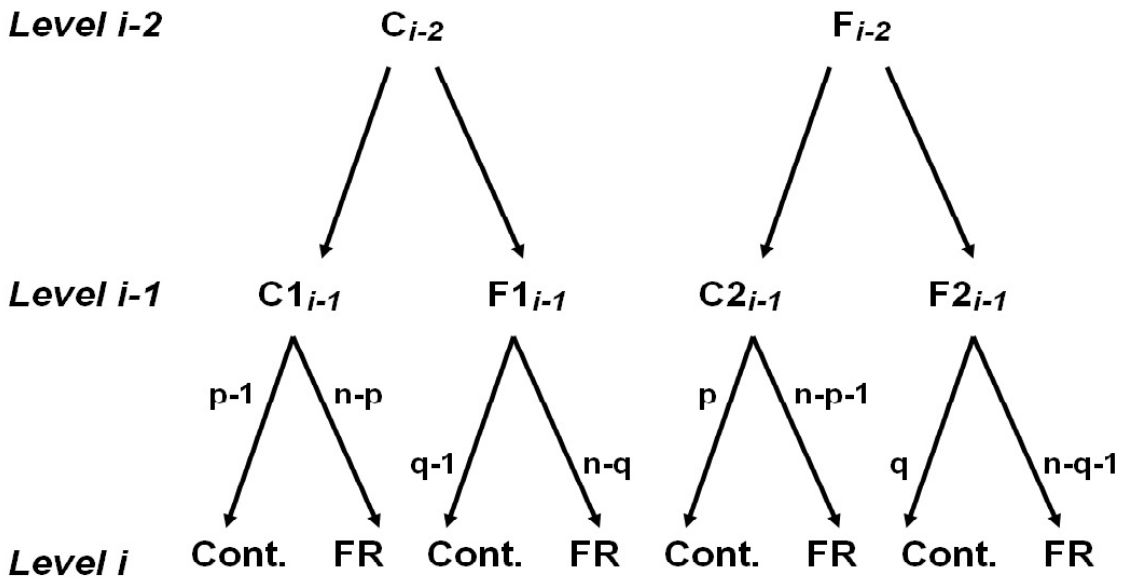

Figure A.1: The relationship between contributors (Cont.) and free riders (FR) at different levels.

of free rider neighbors. Similarly, a free rider peer has q number of contributor neighbors and $n - q$ number of free rider neighbors.

Let X_i denote the number of peers that are i hops away from the querying peer. We also say X_i is the number of peers at level i. X_i can be computed easily.

$$X_i = n(n-1)^{i-1}, \quad i \geq 1 \tag{A.1}$$

Some of these X_i peers are contributors and some are free riders. Let C_i be the number of contributors and F_i be the number of free riders at level i. Thus, $X_i = C_i + F_i$. As we deal with a contributor as the originator of the query, $C_0 = 1$, $C_1 = p$, and $F_1 = n - p$.

We compute C_i in a recursive manner. Figure A.1 shows the relationship between contributors at level $i-2$, $i-1$, and i.

If we assume that C_{i-2} is known then F_{i-2} can be calculated as $F_{i-2} = X_{i-2} - C_{i-2}$.

Upon receiving the query, C_{i-2} number of contributing peers at level $i-2$ will forward it to their contributing neighbors (whose count is denoted with $C1_{i-1}$)

and to their free riding neighbors (whose count is denoted with $F1_{i-1}$) at level $i-1$. Similarly, F_{i-2} number of free riding peers at level $i-2$ will forward the query to their contributing neighbors ($C2_{i-1}$) and to their free riding neighbors ($F2_{i-1}$) at level $i-1$.

As indicated in Figure A.1, we can compute the number of contributors at level i using the number of contributors and free riders at previous levels $i-1$ and $i-2$. Each of the $C1_{i-1}$ contributing peers at level $i-1$ will forward their query to $p-1$ contributors [1]. Then we obtain the following recursive relationship for the number of contributors at level i:

$C_i = C1_{i-1}(p-1) + F1_{i-1}(q-1) + C2_{i-1}(p) + F2_{i-1}(q),$
$C_i = C1_{i-1}p - C1_{i-1} + F1_{i-1}q - F1_{i-1} + C2_{i-1}p + F2_{i-1}q,$
$C_i = p(C1_{i-1} + C2_{i-1}) + q(F1_{i-1} + F2_{i-1}) - (C1_{i-1} + F1_{i-1}).$

We have the following equations:

$C1_{i-1} + C2_{i-1} = C_{i-1}$, and $F1_{i-1} + F2_{i-1} = X_{i-1} - C_{i-1}$; and
$C1_{i-1} + F1_{i-1} = C_{i-2}Y_{i-2}$.

Here, Y_i is the number of neighbors that will receive a query originated or forwarded by a peer i. If the peer is the query originator, i.e., $i = 0$, the number of neighbors to whom the query will be forwarded is n. Otherwise, if the peer is a query forwarder, the number of neighbors to whom the query will be forwarded is $n-1$. In short, if i is 0 then Y_i is n, otherwise Y_i is $n-1$.

Now, the equation that gives the number of contributors at level i becomes:

$$C_i = pC_{i-1} + q(X_{i-1} - C_{i-1}) - Y_{i-2}C_{i-2}, \quad i \geq 2 \tag{A.2}$$

As mentioned before, if the originator of the query is a contributor, $C_0 = 1$ and $C_1 = p$.

As a result, the total number of contributors that will receive the query issued by a contributor is:

[1] We have $p-1$ not p because, those forwarding peers have a contributor parent that is also a neighbor of them.

$$C = \sum_{i=1}^{m} C_i = p + \sum_{i=2}^{m} (pC_{i-1} + q(X_{i-1} - C_{i-1}) - Y_{i-2}C_{i-2}), \quad m \geq 2 \qquad \text{(A.3)}$$

We can use this recursive formula to compute the number of contributors for various settings of the parameters m, n, p, and q. For example, in a P2P network, each peer, a contributor or a free rider, has 2 contributing neighbors and 3 free riding neighbors. That is, $n = 5$, $p = 2$, $q = 2$, and $m = 5$. Using Equation A.3, the number of contributors that a contributing peer's query can reach is computed as 692. If we can control and modify the connections in this network (what we aim with our approach) so that each contributor has 4 out of its 5 neighbors as contributors ($p = 4$), then the number of contributors that will receive the query message issued by a contributor would be 1132. If we can totally isolate free riders, no free rider will have a connection to a contributor and vice versa. This means, p becomes 5, and q becomes 0. In this case, the number of the contributors that will receive the query would be 1706.

These examples show that we can improve the number of contributors in a search horizon of a contributing peer so that the peer can get better search quality. This is the main motivation for our approach.

After searching the network and receiving the query hits, a peer requests download from one of the source peers. However, source peers are subject to high number of download requests and since the upload capacity is limited, they can refuse some of the download requests. Therefore, receiving a query hit does guarantee a successful download.

Assume that on the average a contributor can upload U number of files simultaneously at maximum, and the number of simultaneous download requests that arrive to this contributor is D. Sometimes, contributors can have much more download requests (D) than their upload capacity (U). In that case, when D is larger than U, a contributor will refuse a download request with a probability $P(refuse) = 1 - U/D$. As the ratio of free riders in a P2P network becomes greater than that of contributors, most of these requests will belong to the free riders. As stated above, we aim to reduce the arrival of download requests from free riders. Therefore, we expect a reduction in $P(refuse)$ for the requests coming

from contributors. Hence, we expect an increase in the downloads that contributors can achieve.

Bibliography

[1] K. Aberer and M. Hauswirth, "Peer-to-Peer Information Systems: Concepts and Models, State-of-the-Art, and Future Systems", 18th International Conference on Data Engineering (ICDE), 2002.

[2] K. Aberer and M. Hauswirth, "An Overview of Peer-to-Peer Information Systems", Workshop on Distributed Data and Structures, 2002.

[3] E. Adar and B. A. Huberman, "Free Riding on Gnutella", "http://www.firstmonday.dk/issues/issue5_10/adar", 2000.

[4] N. Andrade, F. Brasileiro, W. Cirne and M. Mowbray, "Discouraging Free-riding in a Peer-to-Peer Grid", IEEE International Symposium on High-Performance Distributed Computing, 2004.

[5] N. Andrade, M. Mowbray, W. Cirne and F. Brasileiro, "When Can an Autonomous Reputation Scheme Discourage Free-riding in a Peer-to-Peer System", Workshop on Global and Peer-to-Peer Computing, 2004.

[6] S. Androutsellis-theotokis and D. Spinellis, "A Survey of Peer-to- Peer Content Distribution Technologies", ACM Computing Surveys (CSUR) Vol. 36, Issue 4, pp. 335-371,December 2004.

[7] A. Asvanund, K. Clay, R. Krishnan and M. Smith, "An Emprical Analysis of Network Externalities in Peer-To-Peer Music Sharing Networks", International Conference on Information Systems, 2002.

[8] A. Atip, K. Clay, R. Krishnan, and M. Smith, "An Empirical Analysis of Network Externalities in Peer-To-Peer Music Sharing Networks", Information Systems Research, 15(2) 155-174, 2004.

[9] Avaki Data Grid Web Site, "www.avaki.com", 2008.

[10] D. Banerjee, S. Saha, S. Sen and P. Dasgupta, "Reciprocal Resource Sharing In P2P Environments", The 4th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05), July, 2005.

[11] BitTorrent Web Site, "www.bittorrent.com/", 2008.

[12] E. Buchmann and K. Bohm, "FairNet - How to Counter Free Riding in Peer-to-Peer Data Structures", The International Conference on Cooperative Information Systems, 2004.

[13] L. Buttyn and J.-P. Hubaux, "Security and Cooperation in Wireless Networks", Cambridge University Press, 2007.

[14] H. Cai, and J. Wang, "Foreseer: A Novel, "Locality-aware Peer-to-Peer System Architecture for Keyword Searches", The 5th ACM/IFIP/USENIX international conference on Middleware, pp. 38-58, 2004.

[15] C. Blake and R. Rodrigues, "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two" , Ninth Workshop on Hot Topics in Operating Systems (HotOS-IX), (Lihue,Hawaii), pp. 1-6, May 2003.

[16] Y. Chawathe, S. Ratnasamy, L. Breslau, and S. Shenker, "Making Gnutella-like P2P Systems Scalable", In Proceedings of ACM SIGCOMM, 2003.

[17] J. Chu, K. Labonte, and B. Levine, "Availability and Locality Measurements of Peer-to-Peer File Systems" Proc. ITCom: Scalability and Traffic Control in IP Networks II Conf., SPIE, vol. 4, 2002.

[18] Clip2, "The Gnutella Protocol Specification v0.4 (Document Revision 1.2)", "http://www9.limewire.com/developer/gnutella protocol0.4.pdf", 2001.

[19] B. Cohen, "Incentives Build Robustness in BitTorrent", Workshop on Economics of Peer-to-Peer Systems, vol. 6, 2003.

[20] C. Cramer, K. Kutzner, and T. Fuhrmann,"Bootstrapping Locality-Aware P2P Networks", The IEEE International Conference on Networks (ICON), 2004.

[21] F. Dabek, E. Brunskill, M. F. Kaashoek and D. Karger, "Building Peer-To-Peer Systems With Chord, A Distributed Lookup Service", Hot Topics in Operating Systems Workshop, May 2001.

[22] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati and F. Violante, "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks", 9th ACM Conference on Computer and Communications Security, November 2002.

[23] N. Daswani and H. Garcia-Molina , "Query-Flood DoS Attacks in Gnutella", ACM Conference on Computer and Communications Security, Washington, DC, November 2002.

[24] N. Daswani, H. Garcia-Molina and B. Yang, "Open Problems in Data-sharing Peer-to-peer Systems", ICDT, 2003.

[25] P. Dewan and P. Dasgupta, "Securing P2P Networks Using Peer Reputations: Is there a silver bullet?", IEEE Consumer Communications and Networking Conference (CCNC 2005) , 2005.

[26] eDonkey Web Site, "http://www.edonkey2000.com", 2008.

[27] eMule Web Site, "http://www.emule-project.net", 2008.

[28] Entropia Web Site, "http://www.entropia.com", 2008.

[29] M. Feldman and J. Chuang, "Overcoming Free-riding Behavior in Peer-to-peer Systems", ACM SIGecom Exchanges, Vol 5, Issue 4, July 2005.

[30] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, "Free- Riding and Whitewashing in Peer-to-Peer Systems", ACM SIGCOMM'04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS), 2004.

[31] G. H. L. Fletcher, H. A. Sheth, K. Brner, "Unstructured Peer-to-Peer Networks: Topological Properties and Search Performance", AP2PC 2004.

[32] Freecast Web Site, "http://www.freecast.org/", 2008.

[33] FreeNet Web Site, "http://www.freenet.com/", 2008.

[34] Free Haven Web Site, "http://www.freehaven.net/", 2008.

[35] F.D. Garcia, and J.H. Hoepman, "Off-line Karma: A Decentralized Currency for Static Peer-to-peer and Grid Networks", 5th Int. Networking Conf.(INC), 2005.

[36] GnuNET Web Site, "http://gnunet.org", 2008

[37] P. Golle, K. Leyton-Brown and I. Mironov, "Incentives for Sharing in Peer-to-Peer Networks", Electronic Commerce, 2001.

[38] Groove Web Site, "http://office.microsoft.com/en-us/groove/", 2008.

[39] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload", ACM Symposium on Operating Systems Principles, 2003.

[40] L. Guo, S. Chen, Z. Xiao,E. Tan,X.Ding, and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems", Proceedings of the 2005 Internet Measurement Conference (IMC'05), 2005.

[41] M. Gupta, P. Judge and M. Amma, "A Reputation System for Peer-to-Peer Networks", Proc. of the 13th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), 2003.

[42] A. Habib, and J. Chuang, "Service Differentiated Peer Selection: An Incentive Mechanism for Peer-to-Peer Media Streaming", IEEE Transactions On Multimedia, vol.8, no. 3, 2006.

[43] D. Hales and B. Edmonds, "Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks", IEEE Transactions in Systems, Man and Cybernetics - Part A: Systems and Humans,Volume 35(3), pp. 385-395, 2005.

[44] M. Ham and G. Agha, "ARA: A Robust Audit to Prevent Free-Riding in P2P Networks", The Fifth IEEE International Conference on Peer-to- Peer Computing (P2P2005), 2005.

[45] J. Han and Y. Liu, "Dubious Feedback: Fair or Not?", Proceedings of the International Workshop on Peer-to-Peer Information Management(P2PIM'06), 2006.

[46] S. Handurukande, A. Kermarrec, F. Le Fessant, L. Massouli, and S. Patarin , "Peer Sharing Behaviour in the Edonkey Network and Implications for the Design of Serverless File Sharing Systems", the First EuroSys Conference (Eurosys'2006), Leuven (Belgium), April 2006.

[47] G. Hardin, "The Tragedy of the Commons", Science, 1968.

[48] D. Hughes, G. Coulson, and J. Walkerdine, "Free Riding on Gnutella Revisited: the Bell Tolls?", IEEE Distributed Systems Online, vol. 6, no. 6, June, 2005.

[49] M. Iguchi, M. Terada and K. Fujimura, "Managing Resource and Servent Reputation in P2P Networks", The 37th Annual Hawaii International Conference on System Sciences (HICSS'04), 2004.

[50] Jabber Web Site, "http://www.jabber.org/", 2008.

[51] M. Jakobsson and A. Juels, "Proofs of Work and Breadpudding Protocols", Proc. Communications and Multimedia Security, September 1999.

[52] M. Jakobsson, A. Juels, "Proofs of Work and Breadpudding Protocols", The Communications and Multimedia Security, 1999.

[53] L. Jian and J. MacKie-Mason, "Why Share in Peer-to-Peer Networks?" , First Workshop on the Economics of Networked Systems (NetEcon06), 2006.

[54] M. Jovanovic, "Modeling Large-scale Peer-to-Peer Networks and a Case Study of Gnutella", Master's thesis, University of Cincinnati, 2001.

[55] M. Jovanovic, F.S. Annexstein and K.A. Berman, "Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella", Technical Report, University of Cincinnati, 2001.

[56] JXTA Web Site, "http://www.sun.com/software/jxta/", 2008.

[57] L. Kahney, "Cheaters Bow to Peer Pressure", "http://www9.wired.com/news/tecnology/0,1282,41838,00.html", 2001.

[58] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "Eigenrep: Reputation Management in P2P Networks", 12th International WWW Conference, 2003.

[59] S. D. Kamvar, M. T. Schlosser and H. Garcia-Molina, "Addressing the Non-Cooperation Problem in Competitive P2P Networks", Workshop on Economics of P2P Systems, 2003.

[60] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "Incentives for Combatting Freeriding on P2P Networks", In Euro-Par, 2003.

[61] M. Karakaya, I. Korpeoglu, O. Ulusoy, "A Distributed and Measurement-Based Framework Against Free Riding in Peer-to-Peer Networks", Proceedings of the IEEE International Conference on Peer-to-Peer Computing (P2P'04), August 2004.

[62] R. Krishnan, Mi. D. Smith, Z. Tang and R. Telang, "The Impact of Free-Riding on Peer-to-Peer Networks", Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 7, January, 2004.

[63] A. Kuzmanovic, D. Dumitriu, E. Knightly, I. Stoica and W. Zwaenepoel, "Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems", the ACM SIGMETRICS'05, 2005.

[64] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the Kazaa Network", The Third IEEE Workshop on Internet Applications, WIAPP 2003.

[65] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit, "Are File Swapping Networks Cacheable? Characterizing P2P Traffic", Proc. of the 7th Int. WWW Caching Workshop, 2002.

[66] C. Li, B. Yu and K. Sycara, "An Incentive Mechanism for Message Relaying in Peer-to-Peer Discovery", Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems, 2004.

[67] Y. Liu, Z. Zhuang, L. Xiao and L. Ni, "AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems", The IEEE Global Telecommunications Conference (Globecom), 2003.

[68] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks" , ICS02, USA, June 2002.

[69] R.T.B. Ma, S.C.M. Lee, J.C.S. Lui, and D.K.Y. Yau,"An Incentive Mechanism for P2P Networks", International Conference On Distributed Computing Systems, Vol 24, pages 516-523, 2004.

[70] E. P. Markatos, "Tracing A Large-Scale Peer To Peer System: An Hour in the Life Of Gnutella", IEEE International Symposium on Cluster Computing and the Grid, 65-74, May 2002.

[71] J. McCoy, "Mojo Nation Responds", "http://www.openp2p.com/pub/a/p2p/2001/01/11/mojo.html", 2001.

[72] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing", TR, HP Laboratories Palo Alto, HPL-2002-57, March 8 , 2002.

[73] J.J.D. Mol, D.H.J. Epema, and H.J. Sips, "The Orchard Algorithm: P2P Multicasting without Free-riding", The 6th IEEE International Conference on Peer-to-Peer Computing, IEEE Society Press, 2006.

[74] MS .NET Web Site, "http://msdn2.microsoft.com/en-us/netframework/", 2008.

[75] Napster Web Site, "www.napster.com/", 2008.

[76] S. Osokine, "Flow Control Algorithm for Distributed 'Broadcast-Route' Networks With Reliable Transport Links", "http://www.grouter.net/gnutella/flowcntl.htm", 2001.

[77] OceanStore Web Site, "http://oceanstore.cs.berkeley.edu/", 2008.

[78] PAST Web Site, "http://research.microsoft.com/ antr/PAST/", 2008.

[79] Peercast Web Site, "http://www.peercast.org/", 2008.

[80] PPLive Web Site, "http://www.pplive.com/", 2008.

[81] Publius Web Site, "http://www.cs.nyu.edu/waldman/publius/", 2008.

[82] L. Ramaswamy and L. Liu, "Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems", Annual Hawaii International Conference on System Sciences, 2003.

[83] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network", The ACM SIGCOMM Conference, 2001.

[84] M. Ripeanu, I. Foster and A. Iamnitchi, "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design", IEEE Internet Computing, Journal special issue on peer-to-peer networking, 6(1), 2002.

[85] J. Ritter, "Why Gnutella Can not Scale. No, Really", "http://www.darkridge.com/ jpr5/doc/gnutella.html", 2001.

[86] C. Rohrs, "Sachrifc: Simple Flow Control For Gnutella", "http://www.limewire.com/developer/sachrifc.html", 2002.

[87] J. Shneidman and D. Parkes, "Rationality and Self Interest in Peer to Peer Networks", Proceedings of the IPTPS Workshop, February 2003.

[88] D. Schoder ,K. Fischbach and C. Schmitt , "Core Concepts in Peer-to-Peer (P2P) Networking", in: Subramanian, R.; Goodman, B. (eds.): P2P Computing: The Evolution of a Disruptive Technology, Idea Group Inc, Hershey, 2005.

[89] H. Schwetman, "CSIM: A C-based, Process Oriented Simulation Language", Winter Simulation Conference, 1991.

[90] SETI@home Web Site, "http://setiathome.berkeley.edu/", 2008.

[91] A. Singh and M. Haahr, "Topology Adaptation in P2P Networks Using Schelling's Model", The Workshop on Emergent Behaviour and Distributed Computing, PPSN-VIII, 2004.

[92] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", Proceedings of the Multimedia Computing and Networking, January, 2002.

[93] S. Saroiu, P. Gummadi, and S. Gribble, "Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts", Multimedia Systems, vol. 9, no. 2, Springer-Verlag, pp. 170-184 , 2003.

[94] K. Sripanidkulchai, "The Popularity of Gnutella Queries and Its Implications on Scalability", www.openp2p.com website, February 2001.

[95] Q. Sun and H. Garcia-Molina,"SLIC: A Selfish Link- based Incentive Mechanism for Unstructured Peer-to-Peer Networks", Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), 2004.

[96] L. Tang, "Identifying Resource Authenticity in P2P Networks", The 2nd International Conference of Applied Cryptography and Network Security (ACNS), 2004.

[97] The Peer-to-Peer Trusted Library Web Site, "http://sourceforge.net/projects/ptptl", 2008.

[98] V. Vishnumurthy and S. Chandrakumar and E. G. Sirer, "KARMA: A Secure Economic Framework for P2P Resource Sharing", Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, June, 2003.

[99] UUSee Web Site, "http://www.uusee.com/", 2008.

[100] B. Yang and H. Garcia-Molina, "PPay: Micropayments for Peer-to- Peer Systems", Proc. 10th CCS, V. Atluri and P. Liu (Eds.), ACM Press, pp. 300-310, New York, 2003.

[101] M. Yang, Z. Zhang, X. Li, and Y. Dai, "An Empirical Study of Free- Riding Behavior in the Maze P2P File-Sharing System", IPTPS, 2005.

[102] B. Yu and M. P. Singh, "Incentive Mechanisms for Agent-Based Peer-to-Peer Systems", Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing (AP2PC 2003), July 2003.