A Comprehensive Performance Evaluation of a Novel Framework Against Free Riding in Peer-to-Peer Networks

Murat Karakaya, İbrahim Körpeoğlu and Özgür Ulusoy

Department of Computer Engineering

Bilkent University

06800 Ankara, Turkey

{muratk,korpe,oulusoy}@cs.bilkent.edu.tr

Abstract

Peer-to-peer computing paradigm has attracted a significant amount of interest as a popular and successful alternative to traditional client-server paradigm for resource sharing and content distribution. However, the existence of high degrees of free riding may be an important threat against the proper operation and high-performance of P2P networks.

In this report, we propose a distributed framework to reduce the degree of free riding and its adverse affects on P2P networks. Our solution primarily focuses on locating free riders and taking counter-actions against them. As part of our solution, we propose an approach in which each peer monitors its neighbors, makes decisions if they are free riders or not, and takes appropriate actions if it decides that one of the neighbors is a free rider. To be able to identify the peers as free riders, we define sample free riding types and describe how they are related with the protocol messages of existing P2P protocols such as Gnutella. As a result, we employ sample formulas to determine if a neighboring peer exhibits any kind of free riding or not. Furthermore, we present example counter action schemes that can be applied to neighbors that are detected as free riders. We then combine the mechanisms to detect free riders and the actions that can be taken against them into a practical framework that can be used in an unstructured P2P network. Unlike other distributed mechanisms against free riding, our framework does not require any permanent identification of peers or security infrastructures for maintaining a global reputation system. In the work, we also discuss the probable attacks to the proposed framework and their effects on it.

We also performed a sample implementation of the proposed framework as part of a simulation model, and conducted extensive simulation tests. Our simulations results show that by reducing the amount of free riding in a P2P network, we can accomplish an increase in the performance of the P2P networks for contributor peers.

Index Terms

Free Riding, Peer-to-Peer Networks, Gnutella, Distributed Computing, Performance Evaluation.

This work is partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) with grant numbers EEEAG-103E014, EEEAG-104E028, and EEEAG-105E065.

I. INTRODUCTION

Peer-to-peer (P2P) computing paradigm has attracted a significant amount of interest for resource sharing and content distribution. Although there are different architectural designs and applications for P2P computing, file sharing is the most commonly used application and in nearly all P2P file sharing systems files are stored at peers, searched through the P2P network mechanisms, and exchanged directly between peers using the underlying network infrastructure and its protocols. In an ideal case, a file that is downloaded by a peer is automatically opened for sharing with other peers. However, peers may be reluctant to share a downloaded file to save their own resources. Therefore, the primary property of P2P systems, the implicit or explicit functional cooperation and resource contribution of peers, may fail and lead to a situation called *free riding*.

As a P2P concept, *free riding* means exploiting P2P network resources (through searching, downloading objects, or using services) without contributing to the P2P network at desirable levels. 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 makes enough contribution to the network by sharing its resources with other peers.

There may be various reasons and motivations for free riding. Bandwidth limitation of peers' connections may be a reason for free riding. Another reason for free riding can be the peers' concern of sharing "bad" or "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 is an important threat against the existence and efficient operation of P2P networks [1] [12]. Thus, free riding causes several negative side effects on P2P networks. In a free riding environment, a small number of peers will serve for a large number of peers. Therefore, many download requests would be directed towards a few serving peers, which may lead to scalability problems [3]. This also leads to a more client-server like paradigm [8], [9] and adversely affects P2P network advantages. For example, fault-tolerance property of P2P networks may be weakened due to the fact that 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. Quality of search process may degrade due to 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 [3], [10]. Moreover, the large number of free riders and their queries will generate a great 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.

Our thesis in this report is that if we can design a framework which detects free riders and takes some counter actions against them, we may 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. By reducing the level of free riding and its effects, we expect to observe performance gains for contributors and performance degradation for free riders. Furthermore, we believe that P2P networks employing the proposed free riding mechanisms would be more robust and scalable.

Our focus in this report is unstructured (pure) P2P networks, specifically Gnutella network [18], and we provide a solution against free riding in such networks. In an unstructured P2P network, there is no central coordination and central indexing mechanism for the shared resources [4], [30], [31]. No peer has a global view of the network, and global behavior of the network emerges from local interactions. These features enable unstructured P2P networks to be very successful, but also bring some problems. Among the problems of such networks is the so-called reputation problem. In an unstructured 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.

In this work, we propose a distributed and localized solution against free riding in unstructured P2P networks, which does not require the storage of any persistent information about peers. Our solution also requires minimal changes to the current protocol processing rules and it does not require any architecture changes.

[1] 1% of the peers provides 37% of the content [1].

Our proposed framework consists of two mechanisms. The first mechanism is for locating and detecting free riders. The second one is for taking discouraging counter actions against them. The mechanisms are distributed and localized, where each peer is only required to monitor its neighbors and make decisions and take actions based on this localized monitoring.

Both the detection mechanism and counter actions are simple, practical and effective. They do not use much resources of contributors either. As opposed to many solutions that execute the counter actions at download request phase, our solution executes the counter actions at 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 helps reducing the network traffic overhead considerably.

We also implemented our solution in a P2P network simulation environment. We first determined some metrics that can be used to evaluate the efficiency and effectiveness of our solution, and we performed extensive simulations to compare a P2P network that is utilizing our framework against a P2P network that is not utilizing our framework. We observed significant performance improvements in a P2P network utilizing our framework.

The organization of the report is as follows. Sections II discusses the related work. In Section III, the mechanisms for locating free riders and taking actions against them are described. In Section IV, we present the simulation model and the performance results. We discuss some possible attack scenarios from free riders to our framework and provide some possible solutions in SectionV. In this section, we also provide simulation results regarding how a P2P system with our framework behaves against malicious attacks of free riders. Conclusions are provided in Section VI.

In Appendix A, we report performance results for various values of important parameters of our simulation model and mechanisms. We also compare our mechanisms with a related work in Appendix B.

II. Related Work

User traffic on Gnutella network is extensively analyzed by Adar ad Huberman in [1], and it is observed that 70% of the peers do not share any files at all, and 73% of peers shares ten or less files. 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 whole query hits in the network.

In a more recent work, Saroiu et al. confirm that there is a large amount of free riding in Gnutella network as well as in Napster [9]. One of the interesting observations they made is that 7% of the peers provide more files than all of the other remaining peers.

Some researchers have attempted to solve the free riding problem by following incentive based approaches. In most of these works, the solution is to apply a pricing scheme for network resources. For example, in [3], Ramaswamy and Liu propose to calculate a utility function for each peer in order to estimate its usefulness to all community. With this method, free riders cannot download files from the system if the utility value is lower than the size of the requested file. However, there are some ways to walk around the utility values. For example, a user can share some small files with fake names resembling popular file names. If other users download these files, that user's utility value will increase. Moreover, the proposed method depends on accurate information about peers and this information is provided by the peers themselves. A P2P network depending on such an approach can be misreported and cheated by rewriting some malicious client programs[2]. Therefore, we think that this method can not fully prevent free riding. Any method proposed to prevent free riding should be designed in such a way that it should not solely depend on user-submitted information, or it should create the right incentives for the peers to report accurate information [20]. Otherwise, free riders may possibly not tell the truth about themselves if they are not given any incentive to do so [9], [21].

In [13], Vishnumurthy et al. suggest using a single scalar value, called Karma, to evaluate a peer's utility to a system like in [3]. The account of a peer is replicated by a group of peers, called the bank-set, in order to secure the Karma against loosing or tampering. The transfer of Karma between peers is executed through bank-set of each peer.

There are some other application domains different than the file sharing where incentive based approaches are applied. For example, in [27], Yu and Singh introduce a new class of P2P network which integrates referral systems and agent-based P2P systems. In the proposed system, each peer is an agent software, and these agents cooperate to search the whole systems

[2]For example, KazaA P2P client program [19] implements a method like the one proposed in [3] in order to prioritize users' requests at source peers. However, a modified version, Kazaa Lite [20], has been released and it maliciously declares its user to be a "Supreme Being" which is the peer with the highest priority (participation level).

through referrals. The authors propose to use pricing mechanisms in these kinds of P2P systems to prevent free riding. In another work [26], Li et al. propose to use an incentive mechanism for preventing free riding in message relaying for P2P discovery. In this mechanism, a peer is rewarded if a service provider is found via a relaying path composed of this peer.

As pointed out in [11], [23], each of above schemes that depend on micro payments have limitations when applied to many common P2P network architectures. In general, incentive schemes based on persistent identifiers are also complicated by the anonymity of peers, collections of widely dispersed peers, and the ease with which peers can modify their online identity [11], [23]. For example for the Karma proposal, to make the scheme work, a group of peers must be known to store Karma value. Whenever a peer's Karma changes, a predefined number of these peers should be reachable. Therefore, the identification of the peers should be known and be permanent. However, unstructured P2P networks do not support permanent and reliable identification mechanisms.

In our work, we do not propose to use any scoring value for a peer's utility to the system. Therefore, we don't have to bother with storing, retrieving, and saving a utility value. At each peer, we just store the information about the neighbors' messages which are routed by the peer itself. Furthermore, we do 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. Our approach can be implemented on both types of P2P networks, i.e., structured and unstructured.

In [28], the authors propose an incentive model, SLIC, to be used in unstructured P2P networks. Their work has some similarities with our mechanisms. Their proposed scheme also depends on the local interaction of peers to encourage cooperation. Each peer assigns weights to its neighbors based on the behavior of the neighbors, and those weights determine the amount of capacity assigned to the neighbors. To assign capacities, each peer counts the number of query hits it receives via its neighbors and updates weights based on these numbers. The weights moderate the future query processing capacity the peer offers to its neighbors. One of the main differences between SLIC and our mechanisms is that while we assess the contribution of each individual neighbor to the monitoring peer and the overall system, SLIC evaluates the contribution of the sub-network reachable via each neighbor. That is, controlled peers are not only responsible for their performance and services but also for their neighbors' performance and services. As

questioned in [29], SLIC simulations are only applied to scenarios in which single ”probe nodes” behave selfishly - nodes do not adapt their behavior to increase their utilities network wide. Consequently, it is not clear how that model would react when most of the peers are acting selfishly rather than just a small number. To answer this question and to compare SLIC with our mechanisms, we adapted and tested SLIC in our simulation environment and provided the simulation results in Appendix B.

III. Our Framework and Mechanisms Against Free Riding

We propose a framework consisting of two mechanisms to create a P2P network environment in which peers will be monitored about their contributions to the P2P network and enforced to act more cooperatively to be able to continue to use the services and resources of the network. The goal of our framework, however, is not to eliminate all possible kinds of free riding entirely from a P2P network. For example, it is not aiming to promote or enforce new content contribution by peers. This may not be feasible either. Our framework aims to improve the current situation and reduce the ill-effects of the free riding problem by detecting free riders and by reducing the amount of service they get from the network. In this way, peers will be indirectly enforced to cooperate and contribute in order to use the services a P2P network provides.

A. Our Approach

Our proposed approach against free riding requires every peer to passively monitor its controlled peers. Two roles are defined for each peer: *monitoring* or *controlled* (see Figure 1). A peer takes both of the 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 (controlled peers). At the same time, the peer is also a controlled peer, which implies that its messages are monitored and recorded by its monitoring peers. By monitoring the messages of its neighbors, a monitoring peer can decide if a neighbor acts as a free rider or not. Upon deciding that the neighbor acts as a free rider, the monitoring peer can take counter-measures against that neighbor to reduce the adverse affects of free riding on the P2P network caused by that neighbor.

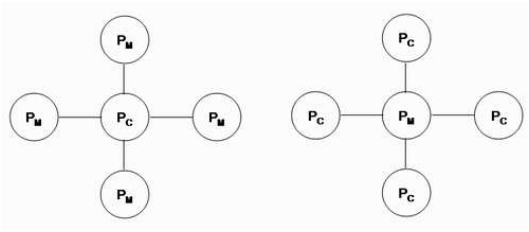

Fig. 1. Peers are in two roles: monitoring and controlled.

Descriptor	Description	Content
`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

TABLE I

TWO MAIN GNUTELLA PROTOCOL DESCRIPTORS (MESSAGES): `Query` MESSAGE AND `QueryHit` MESSAGE.

The statistical information[3] that is to be maintained at a monitoring peer about a controlled peer P consists of a set of counters that are shown in Table II. These counters are updated upon the arrival of protocol messages[4] from the neighbor or upon transmission of protocol messages toward the neighbor. The indications whether the neighbor is free riders or not and the type

[3]Due to the power-law distribution of node degrees observed in P2P networks [6], we expect the average number of neighbors of a peer to be not so large (around 3-4 neighbors on the average), and therefore the overhead imposed by the solution on each peer will not be very large. This implies that the framework is scalable, thanks to its distributed nature.

[4]The messages are implemented with the descriptors in Gnutella protocol [18] (see Table I) , the protocol on which our solution is implemented. Two main Gnutella protocol descriptors (messages) are `Query` and `QueryHit` messages. `Query` message is used for searching P2P network. A peer receiving a Query descriptor will respond with a `QueryHit` if a match is found against its local data set. That is, `QueryHit` is a response to a `Query` message. This descriptor provides the recipient with enough information to acquire the data matching the corresponding `Query` message.

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

TABLE II

OBSERVED DESCRIPTORS

of free riding are then derived from the values of these counters. A different set of counters is maintained for each neighbor.

One issue to consider in our approach is whether there would be enough time during a typical monitoring process to collect enough information about the neighbors to make correct decisions about the behavior of them. In [8], it is stated that about 40% of peers in a Gnutella network leave the network in less than 4 hours, while only 25% of the peers are alive for more than 24 hours. In another work [9] it is reported that the average session duration of both Napster and Gnutella network clients is about 60 minutes. A similar work [10] states that 90% of average session lengths of Kazaa clients is found to be about 30 minutes. All these studies show that we can assume that most peers in a P2P network stay connected long enough so that monitoring peers can collect enough information to make correct decisions.

Another issue to consider is whether a monitoring peer can monitor enough number of messages. In [2], it is reported that the average number of queries received per second for three peers located at three different locations is about 50. The same study also states that around 30 query responses per second are received or originated on the average by a peer. It is also reported that query response ratio per peer is around 10%-12%. This study shows that a monitoring peer will have enough number of messages forwarded over itself to or from a neighbor to judge if the neighbor is a free rider or not.

FR 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 Generation Rate	Normal	Normal	Higher	Normal

TABLE III

SUMMARY OF FREE RIDING TYPES AND THEIR PROPERTIES.

B. Free Riding Types:

In previous works on free riding [12], [13], [14], [23], [24], it is generally assumed that only one type of free riding is exhibited in a P2P network. However, the studies [1], [2], [3], [9], [10] on P2P network traffic and user behavior suggest that not all free riders behave similarly. Therefore types of free riding can be refined further, requiring the definition of different types of free riding with different properties and indications.

In this report we define three free riding types[5]. We believe it is enough at this stage to focus only on three types of free riding in developing a general framework for identification, detection, and prevention of free riding in P2P networks. We also believe that peers exhibiting the free riding types that we focus in this report constitute a large fraction of all kinds of free riders.

A summary of the properties of the proposed free riding types is provided in Table III and the description of each type is given below.

1) Non-contributor: If a peer does not share anything at all or shares uninteresting files, the peer is identified as a non-contributor. A controlled peer P exhibiting this type free riding can be detected by a monitoring peer by counting the `QueryHit` messages (QH_P) originated from the neighbor[6] and comparing the count to the number of `Query` messages (QT_P) sent toward the neighbor (Table II). If the number of `QueryHit` messages received is very few compared

[5]The types of free riding that we define here, however, are not exhaustive. It is possible to define new types of free riding with different properties. In a previous work [22], we had proposed seven different possible free riding types which were combinations of the three types of free riding that we present here.

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

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 originating from a neighbor may indicate that the neighbor is either not sharing any files at all, or is sharing files but the shared files are not interesting and therefore they do not match to the search queries. Unfortunately, a method like this, which is based on counting the `QueryHit` messages, can not be used to distinguish between these two types of reasons of not responding.

Different approaches for setting up a threshold value can be used[7]. Whatever the approach in setting up the threshold is, however, the proposed framework enables a monitoring peer to judge if a neighbor is a non-contributor or not by just observing the neighbor's existing protocol messages, without requiring any new control message to be defined for detection of free riders. Below, we formulate our method to detect a non-contributor 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 remove the warm-up period and to obtain valid statistical information we propose to use a threshold value, τ_{QT}, for the number of forwarded `Query` messages to the controlled peer. A monitoring peer starts deciding about the controlled peer after this threshold is exceeded.

2) Consumer: Peers may contribute to the network with some content, hence they are not non-contributors, but their amount of use of services may be much more than their contribution. It may be useful to further classify such peers in a different class of free riders, called consumers, since what they exhibit is not a desirable behavior either, considering the long term stability of the P2P network and fairness to other peers.

[7]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.

A monitoring peer that would like to identify if a controlled peer P is a consumer or not, counts the `QueryHit` messages that are originated from the neighbor (QH_P) and `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 to use 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 the file. Furthermore, we only count once for all the `QueryHit` messages received for the same query. All `QueryHits` that is received for the same Query will have the same unique Query ID value.

It is not difficult for the monitoring peer to understand if a `QueryHit` message forwarded toward a neighbor is destined to the neighbor or not. This is possible due to fact that in unstructured P2P networks a peer usually sets the initial TTL field value of a `Query` message to a fixed maximum value (max TTL) to limit the flooding of message over the network. Therefore, whenever a peer receives a `Query` message whose TTL is the fixed max TTL, it records to its internal table (using the *message ID* of the `Query` message) that the `Query` is originated from that neighbor[8]. Then, after receiving a `QueryHit` message with the same message ID, the monitoring peer can decide that the `QueryHit` message is for that neighbor and in this way can count the `QueryHit` messages destined to the neighbor.

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 thresholds for QT_P and QH_P counters are used to get rid of the warm-up period before starting making decisions about the behavior of a neighbor.

if ($QT_P > \tau_{QT}$) and ($QH_P > 0$) and

[8]Alternatively we could look to the value of the *hops* field in a `Query` message to see if the message is originated from the neighbor. If the value is equal to zero, the message is originated from the neighbor.

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

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.) further in order to save their connection bandwidth.

In order to detect such a controlled 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

A monitoring peer should count `Query` and `QueryHit` messages that are arriving to itself from the neighbors. It should also count the messages sent or routed from itself to the neighbor.

C. Counter Actions Against Free Riders

When a peer identifies a controlled peer as a free rider, the peer can start taking some counter actions against it. We may think of various possible counter actions that can be applied against a free rider. Here, we will focus on some sample counter actions that can be implemented by just modifying the existing P2P protocols.

Our counter actions, are based on ignoring `Query` messages submitted by free riders or reducing the scope of these queries. In this way we aim to reduce and limit the amount of service that free riders can 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. We think that if we reduce the amount of searching service that a free rider gets, we will also cause a reduction in the amount of downloading service that

the free rider will get. Therefore, our counter actions aim in reducing the propagation of `Query` messages submitted by free riders. Then the free riders will also have less chance of getting `QueryHit` messages and will perform less downloads.

We propose two types of counter action schemes: 1) *single counter action* schemes, and 2) *mixed counter action schemes*. A counter action of type *single*, will apply the same action to all type of free riders. A counter-action of type *mixed*, on the other hand, will apply a different discouraging counter action for each type of free riding. In other words, a mixed counter action scheme will make a distinction between free riders depending on their types while applying discouraging actions.

The proposed single counter actions are described in more detail below[9].

1) Modifying TTL Value : When a peer receives a `Query` message from a controlled peer, it first executes the search on local files for a match, and then forwards the `Query` to its other neighbors. Before the Query message is forwarded, however, its TTL value is normally decremented by one. To act against a controlled free rider, 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 down. This also reduces the overhead imposed by Query messages on the network[10]. 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[11]. We call the corresponding counter-actions TTL-2 and TTL-4, respectively.

2) Dropping Requests: As a sharper counter action, the monitoring peer can simply ignore all the search requests coming from a neighbor that is 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, which is totally different than what happens at the Modifying TTL counter action. We call this counter action `DROP`.

[9]Again, we would like to emphasize that different counter actions may be proposed. The point here is to show that we can accomplish setting up counter actions within the existing protocol definitions and rules.

[10]However, a monitoring peer should be careful about the origin of the `Query` messages while modifying the TTL values of them. It has to modify only the messages that are originated from the neighbor that is a free rider.

[11]Actually, we implemented and observed the effects of different values between 2 and 6 in the simulation experiments, and present the results in Appendix A.

We expect that dropping the search requests of free riders or narrowing down the search horizon of free riders by modifying TTL value does not only punish the free riders, but also significantly decreases the overhead of P2P control messages over the underlying infrastructure. If not controlled, query messages in a flooding based P2P network may become a significant portion of overall network traffic[12]. We believe that decreasing the number of queries submitted by free riders may help improving the performance and scalability of P2P networks and the underlying Internet.

D. A Mixed Counter Action Scheme

A monitoring peer that would like to execute a mixed counter action scheme can apply an appropriate counter action against a free rider depending on the type of free riding. As mentioned earlier, a free riding peer can be either a non-contributor, or a dropper, or a consumer. Then, 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 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 P2P networks. A neighbor that is not identified as a free rider will not get any counter-action.

As stated above, the type of free riding is decided according to the values of statistical counters maintained for a neighbor at the log table of a monitoring peer. When the values of the counters maintained for a neighbor change, the type of free riding decided for the neighbor may also change. For example, if (QH/QT) ratio for a neighbor was first smaller than the respective threshold (i.e., the neighbor is non-contributor), and later becomes greater than that threshold, the neighbor is no longer a non-contributor.

IV. Performance Evaluation

In this section, we first present our simulation model and performance metrics. Then we provide the results of the simulation experiments.

[12]For example, as it is pointed out in [7], an 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, [5] states that the total number of messages including the responses triggered by a single `Query` message can be as large as 26240 (assuming 4 connections per peer).

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 or not.	True	True	False

TABLE IV

PROPERTIES OF PEER TYPES.

A. Overview of the Simulation Model

We used a simulation-based approach to study the model of a typical P2P network with free riding and with our framework integrated, because the model is very complex to study analytically. We implemented our simulation model including our framework on top of the GnuSim P2P network simulation tool that we had developed earlier [25]. GnuSim was implemented as an event-driven simulator using CSIM 18 simulation library [17] and C++ programming language on Windows platform. Interactions between peers and P2P network, such as searching, downloading, pinging, etc., were implemented according to the Gnutella protocol specification given in [18].

Our simulation 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 run.

We assumed that there exist three types of peers in the simulated network: type A, type B, and type C peers. Type A peers and type B peers are contributors (i.e. good peers). Type C peers are free riders. A type C peer which is a free rider can further be classified as a non-contributor, a dropper, or a consumer. A type C peer is randomly and uniformly assigned to one of these 3 types of free riding. Properties of peer types are summarized in Table IV. The properties of each peer type include the population ratio, shared file ratio, maximum number simultaneous uploads possible, query generation mean, and whether peers replicate the downloaded files or not. The default values of each of these properties is set similar to the values reported in [1], [9].

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 distributed to peer groups according to

the type of the groups and the file sharing ratios shows in Table IV. We do not distribute any file to peers that are free riders of type non-contributor or dropper. We assume that each file has 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 submit search queries for them. The inter-arrival time between search requests generated by a peer follows exponential distribution with a mean of 60 time units. We assume that the query generation rate of consumer peers is twice of that of other free riding peers.

Each peer's upload capacity (the number of simultaneous uploads the peer can perform) is limited and is set to 10. If a peer reaches the upload capacity, a new upload request arriving to that peer is rejected by the peer. The requesting peer, can then try to download the file from another peer, if any, selected from a list of peers obtained from the `QueryHit` message. We assume that the requesting peer repeats the same request at most three times. After that the peers gives up with that request and can initiate a request for another file.

Each simulation experiment is run for 2000 units of simulated time. A simulation experiment is repeated 10 times and the results for that experiment are obtained by averaging the 10 individual results.

B. Performance Metrics

In order to measure the performance improvement of our schemes, 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 amount of downloads that can be performed in a P2P system during a fixed time interval for a fixed number of search requests. If peers can download more files from the P2P network, the level of satisfaction they will get from the network will be higher.
- *Number of rejected download requests*: The availability of content and services in a P2P network is an important issue. A network that is providing good service should not reject the contributing peers' requests at high amounts. Since the network resources (bandwidth, storage, processing) are limited, the upload capacity of peers that are contributing to the network will also be limited. If this limit is exceeded, the peers will start refusing download requests.

- *Number of uploads by contributors*: This metric indicates the load imposed on a peer. Contributors can become overloaded due to excessive search and download operations they are involved in. By adapting free riding mechanisms in a P2P system, the load on contributor peers can be decreased by reducing requests served for free riders.
- *Download cost*: We define the download cost for a peer as the ratio between the amount of uploads and the amount of downloads (upload/download) performed by the peer. This ratio indicates the load imposed on a peer compared to the service the peer gets from the network. Such a metric is important to measure how loaded an average peer is compared to the service it gets. The smaller the ratio is, the better it is from the perspective of a peer.
- *Number of P2P network protocol messages*: This metric shows the messaging overhead in the P2P and underlying network. Messaging overhead affect the scalability of a system. Especially in unstructured P2P networks, the messaging overhead may be high due to the flooding approach used in querying. High number of protocol messages sent over the network may also increase the level of congestion in the network, and congestion affects the performance of several network services in various ways, such as causing high delays for remote login applications, increasing query resolution time, decreasing the speed of downloads [16].
- *Fairness*: Fairness metric is used to show 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 adequately compensated with more services. Thus, the solution encourages peers to contribute more and rewards peers based on the extent of their contributions.

C. Simulation Results and Analysis

In our simulation experiments, we first tested the effectiveness of our detection mechanism. Afterwards, we conducted experiments to observe the changes in the performance of a P2P network when counter action schemes are applied[13].

[13]We also executed sensitivity experiments to observe the reaction of the proposed framework against different number of peers, different number of shared files, and different levels of free riding. We observed that the performance results for different parameter settings are consistent and similar with the ones reported in this report. Therefore, due to space limitation, we do not provide those results in the report.

1) Evaluation of Detection Mechanism: The detection mechanism is a crucial part of the framework. Therefore, we did extensive simulation experiments to measure the performance of our framework in detecting free riders and free riding types.

Before discussing the details of the results, we would like to introduce the performance metrics that we have used in evaluating our detection mechanism.

- *Success ratio*: Ratio of the number of peers correctly detected as free riders to the number of peers set as free riders in the beginning of each simulation run[14].
- *Sensitivity ratio*: Ratio of the number 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 the number of peers detected as free riders incorrectly 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. Success ratio is an important metric for single and mixed counter action schemes to operate effectively. On the other hand, sensitivity ratio 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 to the free rider. False alarm ratio is a metric that indicates how many peers are incorrectly detected as free riders, although they are not set as free riders at the beginning of the simulation. If false alarm ratio is high, it means that the framework applies counter-actions to contributors, and contributors are negatively affected from the incorporation of the framework into the P2P network.

An important restriction to the success of the detection mechanism is the behavior and ratio of droppers. This is because the free riders of type dropper 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

[14]Before a simulation run starts, we assign the peers into two main classes: contributing peers and free riders; we further assign the free rider peers into three sub-classes: non-contributors, consumers, and droppers.

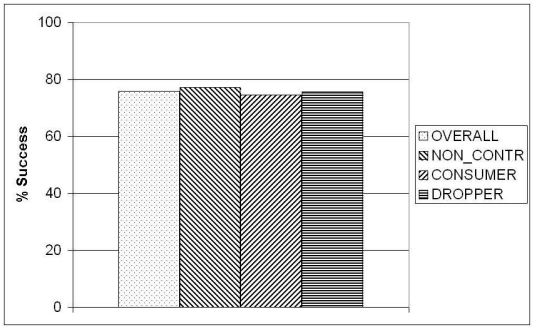

Fig. 2. Success of detection mechanism in detecting free riders and identifying their types of free riding.

success[15]. When τ_{QT} threshold is decreased, however, droppers have more chance to satisfy the threshold value by recording only their queries and may detect free riders. Thus, lowering the value of τ_{QT} increases the success ratio in the presence of droppers, as observed in Table VI.

Figure 2 shows the success of the detection mechanism for default values of simulation parameters. Overall success ratio is about 76%. It means that our detection mechanism is able to detect 76% of peers set as free riders at the start of a simulation run. Droppers are detected with higher success ratio compared to two other free riding types. False alarm ratio is about 9%. That is, 9% of the detected peers is not actually set up as free riders. Their interactions with their neighbors, however, during the simulations lead the detection mechanism to identify them as free riders[16].

In Section III-B, we proposed to use some threshold values for identifying each free riding type. In Table V, the default values of the thresholds are presented. Default values are based on the P2P network traffic observations reported in [1], [2], [8], [10]. As part of our simulations we also tried to observe the effect of setting the thresholds to values in different ranges (Table V).

In Table VI, we observe that when the τ_{QT} parameter is set to lower values, the detection mechanism begins to detect earlier and the success ratio becomes higher. However, false alarm

[15]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. It implies that our framework can not reach a success ratio better than 80% with the current settings of the simulation parameters.

[16]This level of false alarm ratio causes 9% of the peers detected as free riders to face with counter actions. False alarm is the side effect of the detection mechanism. On the other hand, it should be noted that when the performance metrics are observed, we can see that the performance is improved for contributors despite the false alarms (see Section IV-C.2).

Threshold	Description	Default Value	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 V

THRESHOLD VALUES FOR DETECTION MECHANISM.

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 VI

EFFECT OF τ_{QT} THRESHOLD VALUES ON THE DETECTION MECHANISM.

ratio becomes worse with low values of τ_{QT}. This is because we try to decide about a peer with less information available. Therefore, we can say that there is a trade-off between success and false alarm ratios and this trade-off is effected by the τ_{QT} parameter. Sensitivity is not much affected from the value of the τ_{QT} parameter.

Another threshold used in the detection mechanism is $\tau_{non\_contributor}$, which is used to decide if a peer is a non-contributor. In Table VII we provide the results showing the effect of this threshold. An interesting point for this threshold is that 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 it becomes too high. This result suggests us not to use high values 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 under the value 0.01 is almost the same as

$\tau_{non\_contributor}$	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 VII

EFFECT OF $\tau_{non\_contributor}$ THRESHOLD VALUES ON THE DETECTION MECHANISM.

that with the value 0.1. That is, most of the non-contributor peers has the $\frac{QH_P}{QT_P}$ ratio less than the value 0.01. Therefore, the comparison leads to similar success ratio. In Table VII, again we observe that the success ratio is (negatively) correlated with false alarm ratio.

2) Evaluation of Counter-Actions: In Section III-C we proposed two types of counter action schemes in general. We had implemented three different types of single counter action schemes: drop, TTL-4, and TTL-2. We also implemented the mixed counter action. We now report the evaluation of the effectiveness of these schemes. The metrics we used in our evaluation are described in Section IV-B.

Downloads of free riders:

As presented in Figure 3, the number of downloads done by free riders is decreased when the mechanisms against free riding are applied. As discussed in Section III-C, counter-actions against free riders decrease the reach of the `Query` messages sent by peers detected as free riders. This, in turn, reduces the chance of getting a hit to a query submitted by a free rider. In this way, the average number of downloads by free riders is reduced. For example, the dropping counter-action causes about 89% reduction in the number of downloads done by free riders. The least successful counter action is TTL-2 single counter action, which can achieve only 12% reduction. But, still applying a counter-action scheme leads to a lower number of downloads by free riders compared to not applying any counter action against free riders.

The success of the dropping scheme is an expected result, since when dropping all the queries submitted by peers detected as free riders are dropped, those peers can not get `QueryHit` messages back, and therefore they can not download files anymore. They can only download until they get detected. The other schemes, on the other hand, are able to reduce the search

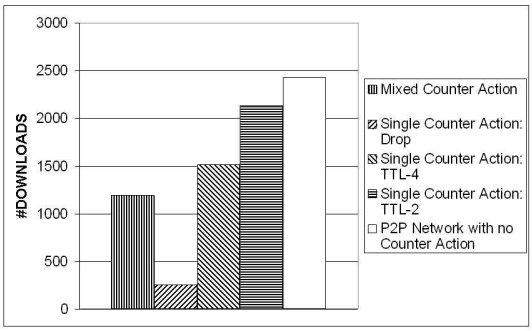

Fig. 3. Decrease in free riding peers' downloads when different counter actions are applied.

horizon of the queries submitted by free riders, but the free riders still have the chance to get `QueryHit` messages and perform 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 that can be done by contributors. Since the capacity of peers uploading content is limited, the download requests of contributors can be rejected sometimes. The rate of rejection is expected to be higher when there are many free riders in the system. Hence eliminating the effects of free riders on P2P network will help in increasing the number of downloads that can be done by contributors. This is indeed shown by Figure 4. As it be derived from the figure, applying our schemes achieves an increase in the number of downloads done by contributors as much as 10%.

An important observation that can be made in Figure 4 is that the improvement in the number of downloads of contributors due to application of single counter actions are not as good as the improvement due to the application of a mixed counter action. While the mixed counter action scheme achieves about 10% improvement, the two single counter actions, TTL-2 and TTL-4, can only achieve about 7% improvement. Dropping counter action scheme, on the contrary, reduces the number of downloads of contributors.

As we discussed earlier, the dropping counteraction prevents the propagation of free riders.

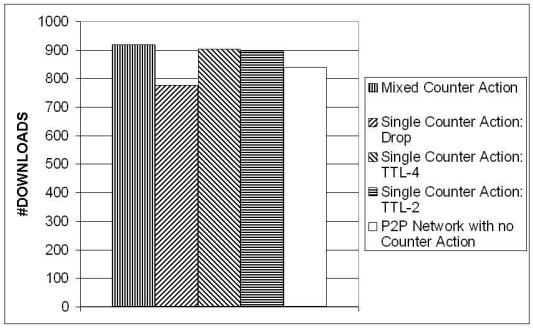

Fig. 4. Increase in contributors' downloads when different counter actions are applied.

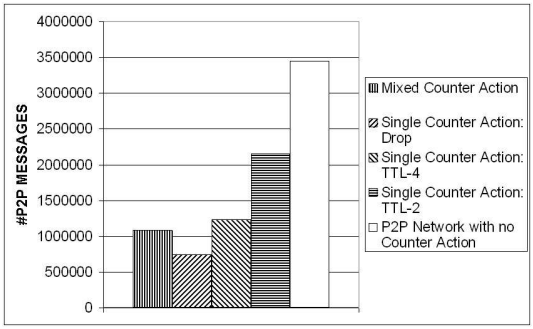

Fig. 5. Decrease in P2P messages of free riding peers when different counter actions are applied.

If, however, some of the peers detected as free riders are not free riders indeed, it means that the scheme punishes contributors and therefore causes the number of downloads of contributors to decrease. As presented in Section IV-C.1, about 9% of detections are actually false alarms. Therefore, when we apply strict counter actions such as dropping, the number of misdetected peers that are negatively affected 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 networks. Counter actions applied against the peers detected as free riders result in a considerable reduction, as much as 78%, in the number of transmitted P2P protocol messages (`Query` and `QueryHit`) originated from and destined to the free riders, as can be see in Figure 5.

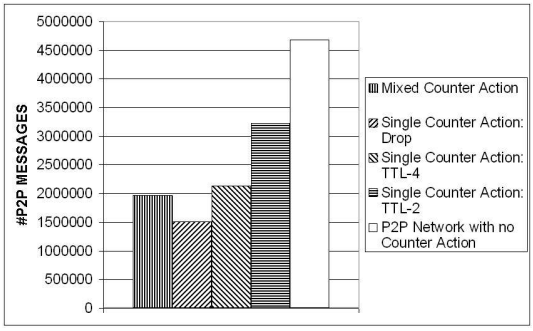

Fig. 6. Decrease in P2P messages of all peers when different counter actions are applied.

When we compare the amount of reductions in the number of transmitted P2P control messages for different counter actions, we see that the dropping single counter action again gives the best results (78%), as it is seen in Figures 5. With the mixed counter action scheme, on the other hand, our framework 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 dropping single counter action scheme leads to a reduction about 68%, whereas the mixed counter action scheme leads to a reduction about 58%. The least successful counter action is TTL-2 in this evaluation, and it only leads to a reduction about 31%. All these results show that applying the proposed framework helps a P2P network to handle more peers with less control messaging overhead and we may argue that 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 of free riders. As we restrict the propagation of the `Query` messages of free riders, we also reduce the amount of `QueryHit` messages destined to the 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 done by the peer in a

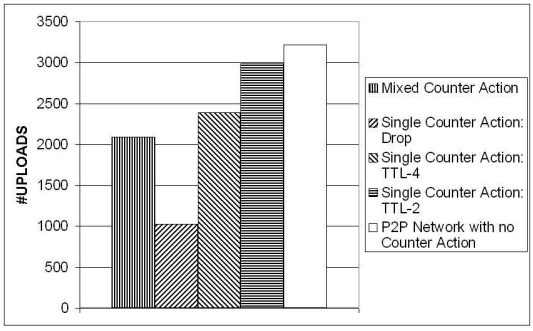

Fig. 7. Decrease in contributors' uploads when counter actions are applied.

given time period. An important goal that we want to achieve with our framework is to reduce the load on contributors. We expect that if we reduce the downloads of free riders, we can also reduce the uploads done by peers, 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 the contributors when a counter action scheme is applied. As can be seen in Figure 7, the scheme that gives the best result is again the dropping single counter scheme, causing a reduction by about 68%. The mixed counter action scheme causes a reduction by about 35%.

Download Cost:

The load on a contributor can also be defined in a different way as a normalized load, which can be defined as the ratio of the amount of uploads to the amount of downloads done by the peer. 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 8, the framework achieves a 65% reduction in the download cost of contributors when dropping single counter action is applied. The framework achieves a 41% reduction when a mixed counter action scheme is applied.

Unsuccessful Downloads:

We also looked to the improvement achieved in the number of unsuccessful downloads done by contributors when the proposed counter action schemes are used. As depicted in Figure 9, the dropping single counter action achieves the best improvement on the metric. The number of unsuccessful downloads are reduced by 97%. Mixed counter action scheme, on the other hand, can reduce 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

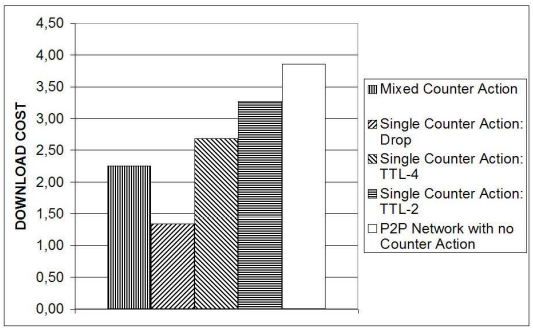

Fig. 8. Decrease in contributors' download cost when counter actions are applied.

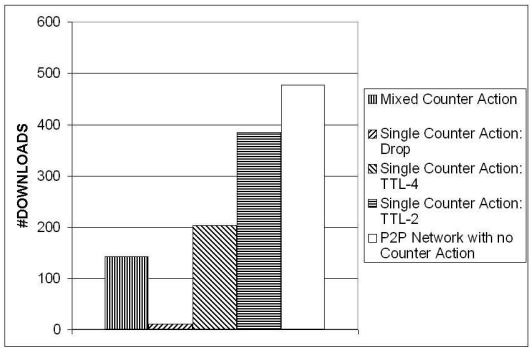

Fig. 9. Decrease in contributors' unsuccessful downloads when counter actions are applied.

downloads may prevent non-free rider peers from accessing files and other resources. When the traffic due to free riders is reduced, then the contributors start reaching to the resources more easily and get better satisfied with P2P network services.

Fairness:

The above results show that when we apply our mechanisms in a P2P network, free riders are faced with decreased number of downloads. On the other hand, contributor peers enjoy an increase in the number of downloads, and a decrease in the number of uploads they need to perform. Furthermore, the download cost of contributor peers reduces. These results clearly show that the proposed mechanisms implicitly help the contributors to get better service by explicitly punishing the free riders.

Before discussing the simulation results about the fairness of the proposed mechanisms, we would like to demonstrate some important properties of the mechanisms related to fairness. We believe that our mechanisms are adaptive and dynamic to peers' behaviors and network's

characteristics. For example, to decide if a controlled peer is a free rider of type Consumer we evaulate the ratio of the number of QueryHit messages provided by the controlled peer to P2P network to the number of QueryHit messages provided by P2P network to that peer (lets call this Consumer-measure). Assume that two peers, Peer A and Peer B, have provided only one QueryHit to the system. Moreover, assume that Peer A received 20 QueryHits for its queries and Peer B received only 5 QueryHits for its queries. If we calculate the Consumer measure for these peers:

Consumer-measure(Peer A): $\frac{1}{20} = 0.05$

Consumer-measure(Peer B): $\frac{1}{5} = 0.2$

As the consumer threshold value is set to 0.1 for the current simulation settings, we would classify Peer A as a consumer ($0.05 < 0.1$) and Peer B as not a consumer ($0.2 > 0.1$), even they both provided the same level of service (one QueryHit) to the system.

A similar example could be given for Non-contributors. The decision if a peer is a free rider of type Non-contributor depends on the ratio of QueryHits-provided to the Queries-received (lets call this Noncontributor-measure). Again, assume that two peers, Peer A and Peer B, have provided only one QueryHit to the system. Moreover, assume that Peer A received 2000 Queries from the system and Peer B received only 1000 Queries from the system. If we calculate the Non-Contributor measures for these peers:

Non-contributor-measure(Peer A): $\frac{1}{2000} = 0.0005$

Non-contributor-measure (Peer B): $\frac{1}{1000} = 0.001$

As the Non-contributor threshold value is set to 0.001 in the experiments, we would classify Peer A as a Non-Contributor ($0.0005 < 0.001$) and Peer B as not a Non-Contributor ($0.001 = 0.001$), even they both provided the same level of service (one QueryHit) to the system.

Our detection mechanism is also adaptive to the changes that happen with time. Assume, for example, that Peer A received 5 QueryHits for its queries and provided only one QueryHit to the system until time t_0. If we calculate the Consumer measure for the peer at time t_0:

Consumer-measure(Peer A): $\frac{1}{5} = 0.2$

We could decide that Peer B is not a consumer ($0.2 > 0.1$). But if Peer A keeps getting service from the system without contributing anymore, the classification would change. Assume that Peer A gets more QueryHits for its new queries, say 15, until t_1, where $t_1 > t_0$. When we evaluate the measure at t_1:

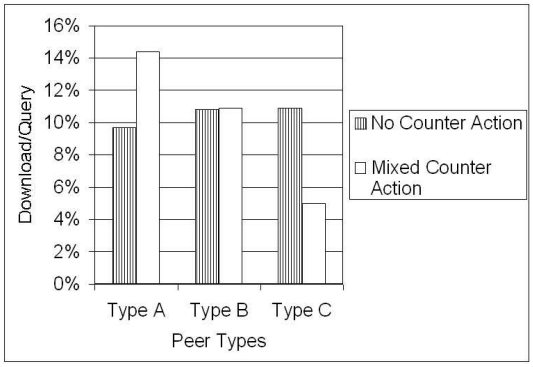

Fig. 10. Changing utility values for different peer types when the Mixed counter action is applied.

Consumer-measure(Peer A): $\frac{1}{5+15}$ = 0.05

At time t_1, it would be classified as a Consumer peer ($0.05 < 0.1$).

As a result, the identification of free riders is not only based on the service that the peers offer, but also on the current P2P system network traffic, the level of service the peers exploit, and the time of the evaluation. This allows the scheme to be more adaptive, dynamic and fair. Thus, our scheme allows different types of contributing peers to receive different levels of service. In the simulation model, we have two types of contributing peers: type A peers and type B peers. Type A peers are less in number (only 10% of all population) but provide 87% of the entire files shared. On the other hand, type B peers are doubled in numbers but they only share 12% of all the shared files. In Figure10, we present the Download/Query ratio results for all types of peers as an indication of their utility from the system. As seen in the figure, when there is no counter action, all types of peers enjoy almost the same level of utility (9-11%). However, when mixed counter action is applied, type A peers get a higher level of service (14%). Type B peers continue to get a similar or a little bit higher level of service. Free riders get a much lower level of service. That is, when mixed counter action is applied, type A peers get higher level of utility compared to type B peers. As stated before, type A peers have more files to share than those of type B peers.

To observe the fairness of our mechanisms, we conducted another set of 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 4, we assigned to the probe peer none(0), 25, 50, 100, and 200 files, and observed the Download/Query ratio.

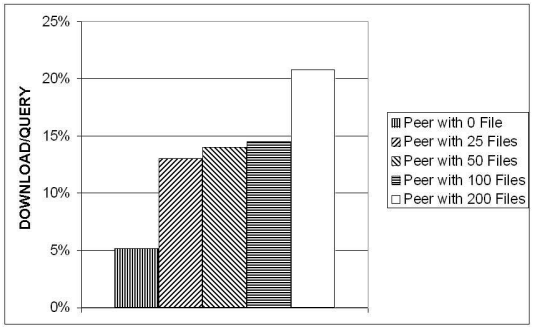

Fig. 11. Increasing utility values for increasing number of files shared by a probe node.

As the figure shows, although the probe peer submits similar amount 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, we can say that, 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, the proposed mechanisms are fair. In other words, a peer contributing more than what is needed to overcome the threshold is adequately compensated. Therefore, the proposed mechanisms not only encourage peers to provide enough services to overcome the threshold barrier, but also encourage them to contribute more to get better service.

V. Possible Attacks to the Framework and Counter Measures

In this section, we discuss a list of possible counter attacks against the free riding prevention mechanisms. We also discuss how we can defend against those kind of attacks.

A. *Fake Query Hit 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. But 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

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 VIII

NEW PROTOCOL DESCRIPTOR

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 feedback mechanism or reputation concept in unstructured P2P networks. To handle this kind of fake QueryHit messages, we propose the use a new descriptor: the Notify descriptor (see Table VIII). The descriptor is used to report about a malicious peer to its neighbor. When a querying peer is refused by a responding malicious peer during download attempt, the querying peer may send a Notify descriptor through the P2P network to reach the 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 to only one 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 on its query routing table[17]. 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 the Notify message, and takes the necessary action against the malicious peer.

There could be some side effects of the proposed Notify descriptor. Some peers may refuse

[17]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. However, we have to make sure that the expiration time of stored Query messages is long enough to accommodate an unsuccessful download attempt and sending of a Notify message.

more connections when they reach the maximum number of connections. Submitting a Notify descriptor for this peer would be unfair and incorrect. To hinder these kinds of false notifications, we propose the use a ratio of the Notify messages to `QueryHit` messages for deciding if a peer is really a malicious peer or not. If that ratio exceeds a predefined threshold, for example 80%, then the peer is identified as a malicious peer and appropriate counter action is invoked.

Another side effect could be occurred by a malicious peer which 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[32], [33], [34], [35]. Some proposals exist in the literature aiming to counter the application-layer DoS attacks[35], [36], [37].

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 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 [35]).

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.

B. 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 incur upload overhead. In that way, free rider peers can conceal themselves. This situation however, can also be prevented by using the `Notify` descriptor as proposed above.

C. Hiding Query Ownership

In the proposed 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.

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 IX

RESULTS OF FREE RIDER MALICIOUS TTL ATTACK (MIXED COUNTER ACTION APPLIED).

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 it submits 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.

But if the free rider sets the TTL to a value greater than the maximum allowed one, this can easily be detected by the monitoring peer. If the free rider sets the TTL to a value less than the maximum allowed one, 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 horizon voluntarily[18].

We observed the effects of this kind of malicious action in our simulations and Table IX provides the results[19].

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

[18]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 looses about 67% of its reach (search horizon) compared to submitting the `Query` with a TTL value of 7.

[19]We have used the mixed counter scheme while performing simulation experiments for evaluating the attacks to the framework

Metric	None of the Peers	Only Non-FRs	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 X

RESULTS OF INSUFFICIENT COOPERATION ATTACK (MIXED COUNTER ACTION APPLIED).

on the system will also decrease considerably if they set the TTL value maliciously. When they cheat on the TTL, they actually reduce the reach of their own queries, and hence the quality of the results they get. As Table IX shows, when a malicious TTL value is used, the downloads of free riders is decreasing. The number of P2P messages observed in the network due to free riders is also decreasing. Hence, acting maliciously on the TTL value does not help much to the free riders. Furthermore, the performance improvements for contributors are still observed even the malicious TTL attack exists. Therefore, we do not see an urgent a need to develop a solution against this kind of TTL attack.

D. Insufficient Cooperation Against Free Riding

Some peers might be reluctant to use the proposed mechanisms against free riding. Especially, free riders would attack the system by disabling the proposed framework. Thus, we may observe low level of cooperation against free riding due the population ratio of free riders. We have simulated such an environment and observed the results.

We would like to compare the case when our proposed mechanisms are applied by only contributors with the case when none of the peers apply the mechanisms. In Table X, we provide the results for both cases.

As expected, the performance is still better compared to the case when our framework is not applied. The improvement can be seen in Table X. As the table 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 no mechanism is applied. This implies that our mechanisms are quite robust against a type of attack where some peers disable the proposed mechanisms in their client software.

E. Constantly Changing Neighbors

It is well known that links between nodes in P2P networks are dynamic. Stated otherwise, peers will be acquiring new neighbors and breaking links with old ones. A free riding peer may constantly change neighbors, and thus may keep utilizing the services without ever being identified as a free rider.

Before, discussing the possible effect of the attack, we would like to remind the related P2P network traffic observations. Related work in this area shows that peers tend to stay connected quite long periods of time. This fact is also stated in Section III-A:

"*One issue to consider in our approach is whether there would be enough time during a typical monitoring process to collect enough information about the neighbors to make correct decisions about the behavior of them. In [8], it is stated that about 40% of peers in a Gnutella network leave the network in less than 4 hours, while only 25% of the peers are alive for more than 24 hours. In another work [9] it is reported that the average session duration of both Napster and Gnutella network clients is about 60 minutes. A similar work [10] states that 90% of average session lengths of Kazaa clients is found to be about 30 minutes. All these studies show that we can assume that most peers in a P2P network stay connected long enough so that monitoring peers can collect enough information to make correct decisions.*"

The reason for peers staying connected for long periods of time is the practical difficulty of disconnecting and connecting again. When a peer disconnects and tries to get re-connected again, there is a chance that the peer will not easily find a node in the network that has enough resources. The capacity of the node that is tried may be low, or the node may be too loaded with network traffic. Therefore, a free rider peer can face difficulties every time it tries to connect to a different peer in the network.

Another reason is that a peer does not get query hit messages immediately after it has submitted a query. There is no benefit of changing neighbors too frequently without waiting for answers to come back. Namely, a peer should not change its neighbors for the time period between

submission of a query and the arrival of the respective query hits (lets call this time interval *search-QueryHit cycle duration*). If the peer would break the existing links too fast, then 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 there will be no meaning 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 does it take to detect the free riders. These depend on several factors.

The success ratio (the ratio of freer riders that are detected correctly) can provide us information about the speed of our detection mechanism. Figure12 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. Figure12 plots the success ratio between time 0 and time 200 (in our current settings, we simulate the network for 2000 time units).

We would like to remind that to remove the warm-up period and to obtain valid statistical information we propose to use a threshold value, τ_{QT}, for the number of forwarded Query messages to the controlled peer. A monitoring peer starts deciding about the controlled peer after this threshold is exceeded.

In Figure12, we observe that, for the current set of simulation parameters, at time 90, 40% of free riders are detected successfully. At time 150, 60% of free riders are detected successfully. If the P2P network traffic becomes higher (i.e. more queries will be forwarded), the time required to exceed the τ_{QT} threshold will be sooner and free riders will be detected faster. 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, according to the current set of simulation parameters. If it changes its neighbors at a rate slower than this, let's say every 100 time units, the chance that it will be detected and face counter-actions is increasing. The probability of detection becomes around 45% for 100 time units.

To investigate the effectiveness of the potential walk-around, we modified our simulation code to also 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

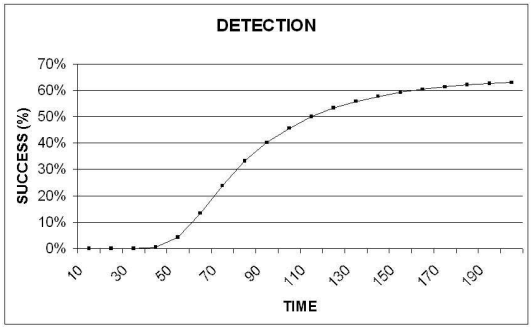

Fig. 12. The Success of the detection mechanism in the first 200 simulation time.

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 the results for two different time intervals between changes of neighbors: 50 and 100 time units. Figure13 shows the results. In the figure, we also included two other utility values. One is the utility value that 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 we have discussed above, when the probe peer changes its neighbors quite frequently, it is expected to get more utility from the system. But when it changes neighbors less frequently, it would probably be detected by its monitoring neighbors and will face a counter-action which will cause its utility to decrease. We remind that the minimum time that the detection mechanism needs to correctly locate the free riders depends on the characteristics of the network traffic. With higher amount of P2P messages, each monitoring peer would have more inputs for the evaluation of its controlled peers within a less amount of time. The point here is the limitation of practical use of the proposed attack.

As can be seen in Figure13, 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 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.

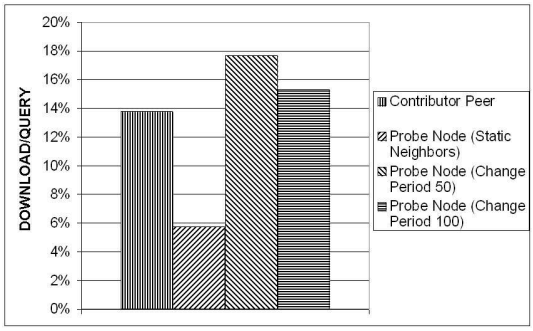

Fig. 13. The results for the Probe peer, when the attack is only applied by the probe FR peer.

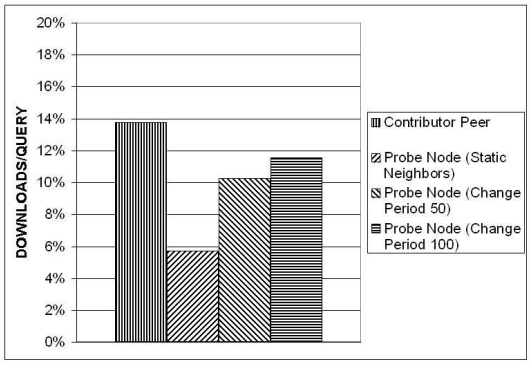

Fig. 14. The results for the Probe peer, when the attack is applied by all the FR peers.

But 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 did experiments for the scenario where all free riders in the network apply the attack hoping to increase the utility they get. Figure14 shows the results. As seen in the figure, the probe acting as a free rider and applying the attack is negatively affected when all free riders in the network apply the attack. In other words, there is not much benefit of applying the attack in this case.

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 loose the connection via these peers and they loose 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 the destined peers. As observed in the figure, this side effect is not negligible. The probe peer lost its advantage considerably when all other free riders also applied the same attack.

Therefore, we may 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 is not increased to a level to justify the practical difficulties of applying the attack. The free rider will not reach a level of utility comparable to that of a contributor peer. Thus, we can conclude that our mechanisms are still effective to discourage free riders and decrease their undesired effects on the P2P network.

F. Increasing Number of Neighbors

A free rider can attack to the proposed mechanisms to increase the benefits it extracts from the network. For example, a free rider with a reduced search horizon can try to search the whole network 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 and they do not increase their benefits very significantly. Below, we would like to provide a simple analysis for the effective of the attack. Later, we share some results of the simulation experiments obtained when the attack applied.

Assume that in a P2P system, average number of connections per peer is 4 and maximum TTL is 7. If a detected free rider employs the attack as suggested above, it can connect to new nodes and but soon after it is also detected by these peers as well. Therefore, its messages' TTL will be decreased always to some value or will totally be dropped. Assume that we implement TTL-4 as a counter action, and a detected FR will be punished by decremented TTL value for its messages, in our example, decremented TTL 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 an 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

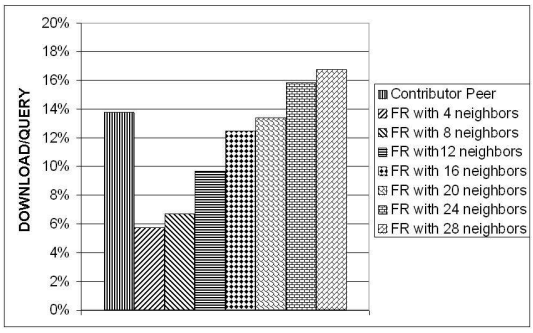

Fig. 15. The results for the Download/Query ratio , when the increased number of neighbors attack is applied by a probe peer.

with TTL 3 (because, as its new neighbors will discover it as a FR sooner). Therefore, each newly connected neighbor can provide at most 17 peers (including itself). To have the same amount of connectivity, FR 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, FR 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 reduce the performance of the peer's system.

We have conducted many experiments to see the effects of the suggested attack. We provide two sets of results. First we collected base information to compare with the results of cheating peer. Therefore, we performed experiments with 2 kinds of randomly selected peers. We first selected a random peer which is a contributor with 4 neighbors. We observed its queries, downloads and the P2P message processing. In Figure15, the results for this peer are given as the first bar from the left. As a second base experiment, we observed a randomly selected free rider with four neighbors. We provided the results for that peer as the second bar from the left. Then, we executed extensive simulations with increasing number of neighbors for the free rider peer.

Figure15 depicts the ratio of Download/Query for each observed peer. Figure 16 shows the average number of P2P messages processed in one unit of simulation time.

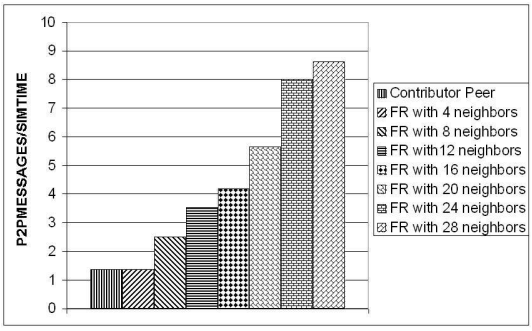

Fig. 16. The results for the P2P Message/Simulation time ratio , when the increased number of neighbors attack is applied by a probe peer.

As seen in Figure 1215, 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 given in the report. 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. In our analytical discussion above, we suggested that a free rider should have 64 times more connection to reach a similar number of connected peers. In simulation experiments, we came up with a less number to have a similar utility value. It is because; in simulation we have less number of peers and less number of connections. But, we can still observe that a free rider peer needs a considerable number of extra neighbors to get a utility from the system in an amount comparable to that a contributor can get.

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 resource 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 16). 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.

VI. CONCLUSION

In this work we have proposed a distributed and measurement based framework to reduce the degree of free riding in unstructured P2P networks. The framework is simple to implement, has low overhead to run, fully complies with concepts and protocols of unstructured P2P networks, and is decentralized to operate efficiently.

We first specified possible free riding types that can be encountered in a P2P network. Then we proposed some mechanisms, which can be used to detect free riders of the defined types. We also proposed some possible counter actions that can be applied against peers detected as free riders.

The mechanisms proposed for reducing the amount of free riding meet the essential requirements of P2P paradigm, such as distributed computing, anonymous connections, unreliable connections, and so on.

By reducing the amount of free riding in a P2P network, we aim to increase the quality of service that peers can get from the network, the availability of content and services, the robustness of the system, the balance of the load on network peers and elements, and the scalability of the network. As the performance results of simulation experiments indicate, the mechanisms manage to reduce free riding and its adverse effects on P2P networks. It is observed that the performance of the P2P network is considerably improved.

In general, we may say that applying "dropping single counter action" against all kind of detected free riders results in better improvements for all the performance metrics except the number of downloads by contributors. We think that this result is due to false detection in determining free riders. As we would like to increase performance for contributors, we offer to use "mixed counter action" scheme. Because it is the best counter action which increases also the downloads of contributors.

We also show that our proposed mechanism can be extended to cope with possible attacks. Furthermore, simulation experiments prove that most of the possible attacks can not render our mechanisms obsolete.

As a future work, we plan to refine the proposed free riding types 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. We finally would like to implement

the proposed mechanism into a Gnutella client and test it on a real P2P network. We expect to observe the effects of proposed mechanisms in real world applications.

REFERENCES

[1] Eytan Adar and Bernardo A. Huberman, "Free Riding on Gnutella", "http://www.firstmonday.dk/issues/issue5_10/adar", 2000.

[2] Evangelos P. Markatos, "Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella", Proceedings of the second IEEE International Symposium on Cluster Computing and the Grid, 65-74, May 2002.

[3] Lakshmish Ramaswamy and Ling Liu, "Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems", 36th Annual Hawaii International Conference on System Sciences (HICSS'03), - Track7,Big Island, Hawaii, January, 2003.

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

[5] Karl Aberer and Manfred Hauswirth, "An Overview of Peer-to-Peer Information Systems", WDAS, 2002.

[6] M. Jovanovic and 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.

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

[8] Matei Ripeanu and Ian Foster and Adriana 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, Volume.6, 2002

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

[10] Krishna P. Gummadi and Richard J. Dunn and Stefan Saroiu and Steven D. Gribble and Henry M. Levy and John Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload", Proceedings of the 19th ACM Symposium on Operating Systems Principles, (SOSP-19), October, 2003.

[11] Ramayya Krishnan and Michael D. Smith and Zhulei Tang and Rahul 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.

[12] Philippe Golle and Kevin Leyton-Brown and Ilya Mironov, "Incentives for Sharing in Peer-to-Peer Networks", Proceedings of the Electronic Commerce'01, 2001.

[13] Vivek Vishnumurthy and Sangeeth Chandrakumar and Emin Gun Sirer, "KARMA: A Secure Economic Framework for P2P Resource Sharing", Proceedings of the Workshop on the Economics of Peer-to-Peer Systems, June, 2003.

[14] Sepandar D. Kamvar and Mario T. Schlosser and Hector Garcia-Molina, "Addressing the Non-Cooperation Problem in Competitive P2P Networks", First Workshop on Economics of P2P Systems, June 2003.

[15] Jeff Shneidman and David Parkes, "Rationality and Self Interest in Peer to Peer Networks", Proceedings of the IPTPS Workshop, February 2003.

[16] Atip Asvanund and Karen Clay and Ramayya Krishnan and Michael Smith, "An Emprical Analysis of Network Externalities in Peer-To-Peer Music Sharing Networks", Proceedings of the 23rd International Conference on Information Systems (ICIS), pp. 15-18, December, 2002.

[17] Herb Schwetman, “CSIM: A C-based, process oriented simulation language”, Proceedings of the 1991 Winter Simulation Conference, pp. 387-396, 1991.

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

[19] Kazaa Web Site, “htpp://www.kazaa.com”, 2004.

[20] Kazaa Lite Web Site, “http://www.k-lite.tk”, 2004

[21] Leander Kahney, “Cheaters bow to peer pressure”, “http://www9.wired.com/news/tecnology/0,1282,41838,00.html”, 2001.

[22] M. Karakaya and I. Korpeoglu and 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.

[23] Nazareno Andrade and Francisco Brasileiro and Walfredo Cirne and Miranda Mowbray, “Discouraging Free-riding in a Peer-to-Peer Grid”, Proceedings of the Thirteenth IEEE International Symposium on High-Performance Distributed Computing (HPDC13), June, 2004.

[24] Nazareno Andrade and Miranda Mowbray and Walfredo Cirne and Francisco Brasileiro, “When Can an Autonomous Reputation Scheme Discourage Free-riding in a Peer-to-Peer System”, Proceedings of the 4th Workshop on Global and Peer-to-Peer Computing(GP2PC), April 2004.

[25] Murat Karakaya, Ibrahim Korpeoglu, Ozgur Ulusoy, “GnuSim: A Gnutella Network Simulator”, Technical Report BU-CE-0505, Department of Computer Engineering, Bilkent University, 2005, “http://www.cs.bilkent.edu.tr/tech-reports/2005/BU-CE-0505.pdf”.

[26] Cuihong Li and Bin Yu and Katia 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.

[27] Bin Yu and Munindar 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.

[28] Qixiang Sun and Hector 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.

[29] 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.

[30] George H. L. Fletcher, Hardik A. Sheth, Katy Brner, “Unstructured Peer-to-Peer Networks: Topological Properties and Search Performance”, AP2PC 2004.

[31] Schoder D., Fischbach K., Schmitt C., “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.

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

[33] Daswani, Neil and Garcia-Molina, Hector and Yang, Beverly, “Open Problems in Data-sharing Peer-to-peer Systems”, ICDT, 2003.

[34] F. Dabek and E. Brunskill and M. F. Kaashoek and D. Karger, “Building peer-to-peer systems with Chord, a distributed lookup service”, Proc. 8 Wshop. Hot Topics in Operating Syst.(HOTOSVIII), May 2001.

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

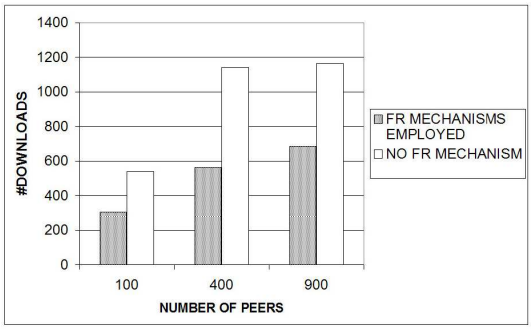

Fig. 17. Decrease in free riding peers' downloads for different numbers of peers in the network.

[36] S. Osokine, "Flow control algorithm for distributed 'broadcast-route' networks with reliable transport links "http://www.grouter.net/gnutella/flowcntl.htm", 2001.

[37] Christopher Rohrs, Sachrifc , "Simple flow control for gnutella "http://www.limewire.com/developer/sachrifc.html", 2002.

APPENDIX A: EFFECT OF FACTORS

We conducted some more performance experiments to evaluate the impact of some critical parameters. Here, we present and discuss the performance results obtained for different values of two important parameters: 1) number of peers in the network, 2) ratio of free riding peers in the network. The number of peers in the network was varied in a range between 100 to 900, and the free riding population ratio was varied in a range between 60% to 90%.

The Number of Peers

We first investigated how the number of peers affects the performance results. To save space, we present here only the results for three metrics: downloads of free riders, downloads of contributors, and P2P message count transmitted in the network. Figures 17, 18, 19 show the results we obtained for these metrics for 3 different values of number of peers: 100, 400, and 900.

From the figures we observe that the number of peers does not affect the improvements achieved with our mechanisms. For all three different number of peers, we obtain a decrease in the downloads of free riders by about 50%. Since the mechanisms are distributed in nature, this is an expected result.

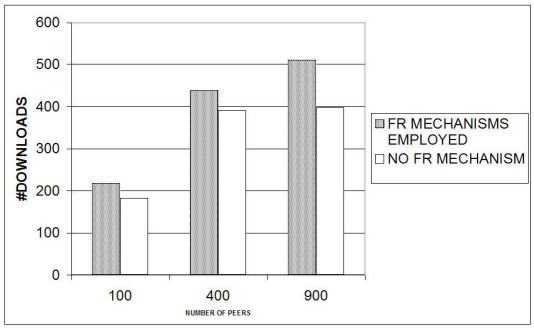

Fig. 18. Increase in contributors' downloads for different number of peers.

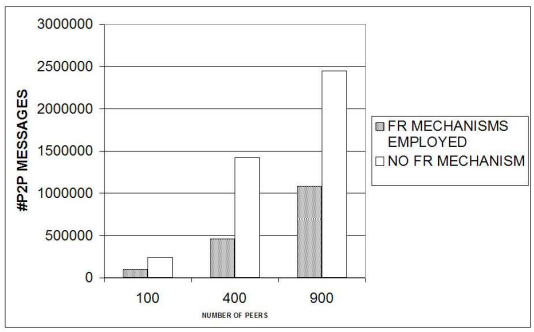

Fig. 19. Decrease in P2P messages transmitted in the network due to free riders for different number of peers in the network.

The number of peers also does not affect the increase in the number of downloads of contributors. No matter what the peer population is, contributors enjoy an increase in their number of downloads as much as 28% (Figure 18).

As seen in Figure 19, the number of P2P protocol messages transmitted in the network decreases almost to the half for different sizes of peer population when our framework is applied. That is, the number of P2P protocol messages in a 900-peer network when our framwork is employed is almost equal to that of a 400-peer network when our framework is not employed. This indicates that our framework supports the scalability of P2P networks with respect to the number of peers and their messages.

The Size of Free Rider Population

We also observed the effect of the size of free rider population in three metrics mentioned above. As seen in Table XI, regardless of the ratio of free riders, our framework achieves a

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

TABLE XI

EFFECT OF FREE RIDER POPULATION ON THE NUMBER OF FREE RIDERS' DOWNLOADS.

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

TABLE XII

EFFECT OF FREE RIDER POPULATION ON THE NUMBER OF CONTRIBUTORS' DOWNLOADS.

reduction in the number of downloads of free riders. For a smaller ratio 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 XII, the increase in the number of downloads by contributors reaches up to 36%.

Table XIII shows the effect of free rider population ratio to the messaging 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%.

FR Popul. (%)	# P2P msgs with Frmwrk	# P2P msgs without Frmwrk	Change(%)
60%	973189	1994822	-51%
70%	826705	1932032	-57%
80%	768145	1706333	-55%
90%	711429	1736857	-59%

TABLE XIII

EFFECT OF FREE RIDER POPULATION ON THE NUMBER OF P2P MESSAGES OF ALL PEERS.

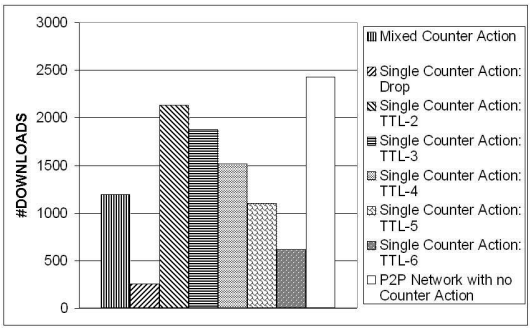

Fig. 20. Downloads of Free Riders when different counter actions are employed.

Modifying TTL

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 20 and 21, 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 20, TTL-2 has the least effect while TTL-6 is most effective in reducing the download of FRs. We also provide the results in terms of the reduction in the number of P2P messages of FRs in Figure 21.

The level of the effect of the 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, we observed that TTL-2 had the least improvement effect 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 the modifying TTL action on the system performance, we select TTL-2 and TTL-4

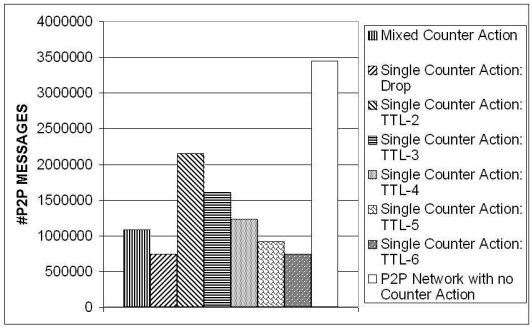

Fig. 21. The Number of P2P messages of Free Riders when different counter actions are employed.

as representative values in this report.

APPENDIX B: COMPARING THE PROPOSED FRAMEWORK WITH ANOTHER APPROACH

As mentioned in Section II, Sun and Garcia-Molina have proposed an incentive model, called SLIC, for promoting cooperation in unstructured P2P networks [28]. The SLIC model has similarities, but also some differences with our proposed mechanisms. In this section, we provide the results of a comparison of SLIC and our scheme. The results were obtained in our simulation environment in which we implemented an adapted version of SLIC as well. Experimental results are summarized in Table XIV.

In SLIC paper [28], simulations are performed only for scenarios where only a single probe node is selected in a P2P network to act selfishly and as a free rider. Consequently, it is not clear how the model would react when most of the peers would become free riders rather than just a small number [29]. In our simulation experiments, however, we obtained SLIC results also for scenarios where there exists a prevalent free riding in the simulated P2P network and where all peers apply the SLIC mechanism.

The first important result that we obtained in our simulation environment about SLIC is that, SLIC is not very effective in differentiating or correctly identifing contributors and free riders in a P2P network. Therefore, it causes a decrease in the downloads of not only free riders, but also contributors; hence in the downloads of all peers no matter they are free riders or not. The amount of downloads that can be performed by free riders and contributor peers is decreased by about 43% and 48%, respectively. This result is not acceptable for contributors. Our scheme, however, does not cause a reduction in the downloads of contributors, except when the DROP

Metric	Mixed	Drop	SLIC
# Downloads of FRs	-50.70%	-89.40%	-43.30%
# Downloads of non-FRs	9.52%	-7.38 %	-48.50%
# P2P Messages of FRs	-68.50%	-78.31 %	-55.30 %
# P2P Messages of all peers	-57.82 %	-67.77%	-55.30 %
# Uploads of non-FRs	-34.89%	-68.10%	-44.30 %
# Unsuccessful Downloads of non-FRs	-70.23%	-97.69 %	-89.70 %

TABLE XIV

PERFORMANCE COMPARISON RESULTS OF OUR COUNTER-ACTIONS AND SLIC. NEGATIVE VALUE INDICATES REDUCTION COMPARED TO NOT APPLYING ANY SCHEME.

counter-action is applied[20]. Even when we compare the SLIC mechanism to our drop counter-action mechanism, as reported in TableXIV, the drop counter-action performs better with respect to almost all metrics in the table. When we compare SLIC to the mixed counter-action, on the other hand, apparently the mixed counter-action provides better results than SLIC, as far as the metrics in the table are concerned.

There can be several reasons why SLIC can not differentiate well between free riders and contributors. One of these reasons is the fact that each node applying SLIC monitors the number of all query hits it receives via its neighbors and updates weights which moderate the future query processing capacity it offers to others. That is, SLIC evaluates the contribution of the sub-network reachable via each neighbor to decide on the amount of service to be provided to these neighbors. In our proposed framework, however, we assess the contribution of each individual neighbor to the system. In other words, in SLIC, controlled peers are not only responsible for their performance and services, but also for the performance and amount of contributions of their neighbors. Considering the prevalence of free riders, even the contributors can not provide a better service than the services provided by all the peers connected through them. Because, even a peer increases its contribution, it can not have any control over the other peers to increase their contributions. At the end, in SLIC, the monitoring peers punish or reward the controlled peers

[20]We observed a negative effect (-7%) on good peers, because the drop counter-action applies a sharp counter action to all kinds of free riders, and this affects the contributors in a negative way when false alarms are triggered.

not only according to their contribution but also according to the contribution of all the peers connected through them. We executed several experiments to observe the effect of increased individual contribution on the performance of a single peer. We selected a peer randomly, and increased its shared file size in each experiment. However, we could not observe any meaningful increase in the service it gets. This observation is in line with the above arguments.