Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 240–256
computer communications
www.elsevier.com/locate/comcom

A connection management protocol for promoting cooperation in Peer-to-Peer networks ☆

Murat Karakaya, İbrahim Körpeoğlu *, Özgür Ulusoy

Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey

Available online 16 August 2007

Abstract

The existence of a high degree of free riding in Peer-to-Peer (P2P) networks is an important threat that should be addressed while designing P2P protocols. In this paper we propose a connection-based solution that will help to reduce the free riding effects on a P2P network and discourage free riding. Our solution includes a novel P2P connection type and an adaptive connection management protocol that dynamically establishes and adapts a P2P network topology considering the contributions of peers. The aim of the protocol is to bring contributing peers closer to each other on the adapted topology and to push the free riders away from the contributors. In this way contribution is promoted and free riding is discouraged. Unlike some other proposals against free riding, our solution does not require any permanent identification of peers or a security infrastructure for maintaining a global reputation system. It is shown through simulation experiments that there is a significant improvement in performance for contributing peers in a network that applies our protocol.

Keywords: Peer-to-Peer networks; Free riding; Connection management; Distributed systems

1. Introduction

Free riding is an important threat against efficient operation of Peer-to-Peer (P2P) networks. In a free-riding environment, a small number of contributing peers serve a large number of peers; many download requests are directed towards a few sharing peers. This situation may lead to scalability problems [3] and to a more client-server-like paradigm [5,6], which overweigh the benefits of P2P network architecture. Additionally, renewal or presentation of interesting content may decrease in time, and the number of shared files may grow very slowly. The quality of the search process may degrade due to an increasing number of free riders on the search horizon. Moreover, the large number of free riders and their queries generate an extensive amount of P2P network traffic, which may lead to degradation of P2P services and inefficient use of the resources of the underlying network infrastructure.

There are various reasons for free riding. Bandwidth limitation of peers' connections may be one reason. Another reason might be peers' concern about sharing "bad" or "illegal" data from their own computers, even though they are not concerned about using this type of data. Some peers may also have security concerns when they share resources.

In this paper, we propose a connection-based solution against free riding that will alleviate the problems associated with free riding. Our solution involves the definition and use of two new connection types (IN and OUT connections) and a P2P Connection Management Protocol (PCMP) that dynamically establishes the connections between peers, and adaptively modifies the P2P topology in reaction to the contributions of peers. Our protocol promotes cooperation among peers and discourages free riding, and can be used in unstructured P2P networks such as Gnutella [10]. Our claim is that if we can adjust the

☆ This work is partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) with Grant Nos. EEEAG-104E028, and EEEAG-105E065.
* Corresponding author. Tel.: +90 312 2902599; fax: +90 312 2664047.
E-mail address: korpe@cs.bilkent.edu.tr (İ. Körpeoğlu).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.08.010

P2P network topology dynamically in reaction to peers' contributions, the adapted topology can favor the contributing peers in getting service from the P2P network. The adapted topology can also exclude the free riders from the P2P network and therefore the adverse effects of free riding can be reduced as well. Furthermore, we expect that our approach will help a P2P network to become more scalable and robust. We did extensive simulations to evaluate our protocol and we have seen significant improvement in the performance of a P2P network with free riders when our solution is applied.

The organization of the paper is as follows. In Section 2, we discuss the related work. In Section 3, we describe our solution and the PCMP connection management protocol. In Section 4, we present our simulation model and provide the simulation results. In Section 5, we discuss some possible attacks to our scheme and how we can cope with them. Finally, in Section 6, we give our conclusions.

2. Related work

User traffic on the Gnutella network was extensively analyzed by Adar and Huberman in [1], and it was reported that 66% of the peers do not share any files at all, while 73% of them share ten or fewer files. Furthermore, 63% of the peers who share some files do not get any queries for these files; and 25% of all peers provide 99% of all Query Hits in the network.

Saroiu et al. confirm that there is a lot of free riding in Gnutella as well as in Napster [6]. They observed that 7% of the peers provide more files than all of the other peers combined.

In a recent work [19] Hughes et al. pointed to an increasing downgrade in the network's overall performance due to free riding. Their results indicated an increasing level of free riding compared to Adar and Huberman's work. For example, they observed that 85% of peers share no files at all. They concluded that free riding was becoming more prevalent.

In another work, Yang et al. reported their findings about free riding behavior in the Maze P2P system [20]. They also found a high level of free riding, with about 80% of the peers behaving like clients. They observed that client-like users (free riders) were responsible for 51% of downloads, but for only 7.5% of uploads. These statistics suggest the existence of free riding in spite of the incentive mechanism provided by the Maze P2P system.

All these observations have caused researchers to be concerned about the free riding problem and to propose solutions. In fact, some mechanisms against free-riding have already been implemented ([20–23]). There are also a number of solutions that have been proposed in research studies ([3,7,8,11,14,24,25]).

Existing mechanisms and proposed solutions for the free-riding problem can be categorized into two main groups: (a) incentive-based and (b) reciprocity-based schemes.

Incentive-based solutions have been proposed to encourage user cooperation within P2P systems. One of the most common way of implementing incentives is to apply telecommunications models for pricing network resources by incorporating micro-payments in P2P networks, such as KARMA [8], ARA [24], PPAY [26], etc. In these systems, each user has to *purchase* service on demand, using a virtual *currency* that is obtained as payment for providing service in turn. Some other incentive-based approaches implement reputation mechanisms [25,27,28]. Reputation-based approaches depend on identifying and monitoring peers' contributions to other peers, and then refusing service to peers with *bad* reputations.

The 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 complicated by the anonymity of peers, by collections of widely dispersed peers, and by the ease with which peers can modify their online identity [7,12].

Reciprocity-based schemes have been proposed as non-monetary mechanisms based on reciprocity among peers, such as [3,11,14]. Peers maintain histories of past behavior of other peers and use this information in their decision making processes. These schemes can be based on direct reciprocity (Tit-for-Tat) or indirect reciprocity (Utility-Based). In direct reciprocity schemes, peer A decides how to serve peer B based solely on the service that B has provided to A in the past. In contrast, in indirect reciprocity schemes, the decision of A also depends on the service that B has provided to other peers in the system. However, there are some ways of getting 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. Additionally, relying on information about a peer that is stored and provided by the peer itself may cause problems as well [6].

In [14], the authors propose an incentive model to encourage cooperation in unstructured P2P networks. This model, called SLIC, depends on the local interactions of peers. In SLIC, each peer assigns weights to its neighbors and updates these weights based on the number of Query Hits it receives via each neighbor. Those weights determine the amount of messaging capacity assigned to each neighbor.

In a previous work [11], we also proposed a framework which focuses on detection of neighbors that are free riders and taking counter actions against them. The proposed framework counts both query hits and query messages, and considers the originator and receiver of these messages. Based on this information, peers make a decision about their neighbors. The proposed framework also categorizes the free riders into several categories. This enables the framework to apply several different counter-actions that are tailored to different types of free riding. The framework assesses the contribution of each neighbor both to the monitoring peer and to the overall system.

Our proposal in this paper, the P2P Connection Management Protocol (PCMP), is another solution to the free riding problem with an approach that is quite different than the methods mentioned above. The PCMP protocol is based on managing connections among peers to discourage free riding and to provide incentives for cooperation. The scheme is distributed and does not require a central entity to control and coordinate. It uses a new connection type to connect peers together. The new connection allows the requests (queries) to be passed in only one direction. Our scheme manages those types of connections so that, eventually, contributors become more close to each other in the network, and free riders become isolated.

There exist some other studies which also focus on modifying P2P topology such as [16–18,29,30]. However, these works aim to solve the topology mismatching problem and improve the search quality; they do not attack the free riding problem directly. In [16], Liu et al. proposed a solution called the Adaptive Overlay Topology Optimization (AOTO) to optimize inefficient overlay topologies for improving P2P search and routing efficiency. In another work [17], Crameret et al. also aimed to create a topology refinement by modifying the bootstrapping mechanism in the P2P network. In [18], Singh and Haahr proposed to modify the P2P network topology so that peers with similar properties become close to each other. Similarly, in [29], Cai and Wang proposed a two-layer (neighbors and friends) unstructured P2P system for better keyword searches. The neighbors overlay is created according to network proximity while the friends overlay is built according to the online query activities. In order to increase the search quality, they try to avoid the free riders in the system while routing the queries. Primarily, the friend overlay is used to route the queries. Because, the friends overlay is constructed in such a way that free riders can not be friends of any peer. However, in their system any peer, including free riders, may issue queries to the system which allows free riders to use the network resources. Chawathe et al. focused on scalability problem in unstructured P2P networks and applied dynamic topology adaptation [30]. They specifically aimed to match the query capacity of the peers with the routed queries to avoid the peers become overloaded by high query rates.

3. P2P connection management protocol

In this section, we first describe our motivation and highlight the benefits of our approach through a simple analytic evaluation. We then give the details of our two new connection types and the connection management protocol, that are proposed to control the connections between contributors and free riders.

3.1. Our approach and motivation

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

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

Second, we propose a P2P Connection Management Protocol (PCMP) to establish and release these two types of connections. The protocol considers the peer contributions while establishing and releasing connections. Hence free riders can be disconnected from contributing peers and even get isolated sometimes. In this way, the associated problems with free riding can be alleviated. Moreover, contributing peers may establish connections to not free riders, but to other contributors and therefore the number of contributors in their search horizon can be increased. Thus, contributors can have better chance to get Query Hits and downloads.

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

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

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

We now provide a motivational example about how we can improve the performance in terms of some of these metrics in a P2P network using our protocol.

The probability of getting a Query Hit depends on many factors including the popularity of the requested file, the number of files shared by peers, and the number of contributing peers in the search horizon. If we assume even popularity and even number of shared files by each peer, then the number of contributing peers in the search horizon will

be the factor determining the hit probability of a query. Therefore, increasing the number of contributors in the search horizon is important for receiving better service from the P2P network.

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

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

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

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

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

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

Upon receiving the query, C_{i-2} number of contributing peers at level $i-2$ will forward it to their contributing neighbors (whose count is denoted with $C1_{i-1}$) and to their free riding neighbors (whose count is denoted with $F1_{i-1}$) at level $i-1$. Similarly, F_{i-2} number of free riding peers at level $i-2$ will forward the query to their contributing neighbors ($C2_{i-1}$) and to their free riding neighbors ($F2_{i-1}$) at level $i-1$.

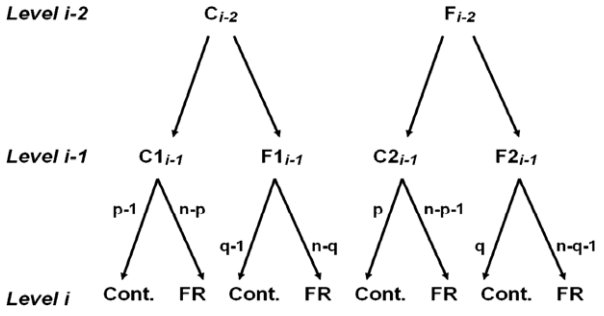

Fig. 1. The relationship between contributors (Cont.) and free riders (FR) at different levels.

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

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

We have the following equations:

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

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

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

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

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

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

$$C = \sum_{i=1}^{m} C_i$$
$$= p + \sum_{i=2}^{m} (pC_{i-1} + q(X_{i-1} - C_{i-1}) - Y_{i-2}C_{i-2}), \quad m \geqslant 2 \tag{3}$$

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

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

and q becomes 0. In this case, the number of the contributors that will receive the query would be 1706.

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

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

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

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

3.2. A new connection type: One-way request connections

In the current unstructured P2P networks like Gnutella, a connection established between a pair of peers is used to exchange all types of P2P protocol messages in both directions including Queries, Query Hits, Pings and Pongs (Fig. 2). PCMP modifies this assumption by proposing a new P2P connection type called *One-Way-Request Connection* (OWRC). As seen in Fig. 3, an OWRC between two peers is still a TCP connection and can carry messages in both directions. However, there is a restriction on what types of messages can be carried in which direction of the connection. The connection is called one way because it can transfer requests in only one direction. In other words, over any OWRC the requests (Query, Ping) can only travel in one direction and the replies (Query Hit, Pong) can only travel in the other direction. Such a connection cannot be used to send and receive all kinds of protocol messages in both directions at the same time. The restrictions on the type of messages and their directions are enforced at the application level by PCMP.

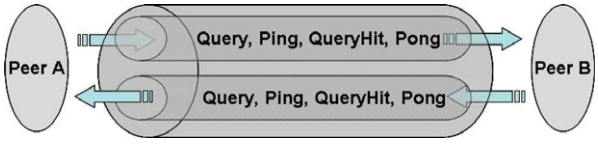

Fig. 2. A general P2P connection between two peers, which enables both of them exchange all types of P2P messages.

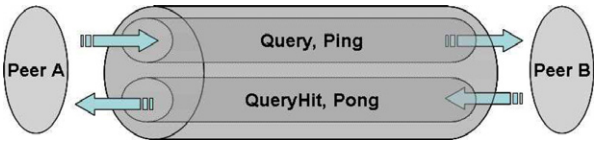

Fig. 3. An OWRC between two peers, which limits the direction and the types of P2P messages exchangeable.

In Fig. 3, one end of the OWRC can be considered a *requester* (Peer A) and the other end as a *responder* (Peer B). The requester sends Query and Ping messages and receives the corresponding Pong and Query Hit messages via the OWRC. A responder, on the other hand, receives Query and Ping messages and replies with Query Hit and Pong messages through the same OWRC. In the rest of the paper, we will call such an OWRC an *OUT-connection* at the requester end and an *IN-connection* at the responder end. Hence, in Fig. 3, peer A has an OUT-connection and peer B has an IN-connection. We will also say that peer A has an *OUT-connected peer*, which is peer B. And peer B has an *IN-connected peer*, which is peer A.

If we would like to transfer requests from the other direction as well, from B to A, we need to establish another OWRC directed from B to A as depicted in Fig. 4. However, we stress again that these connections are logical and can be implemented on top of either one or two TCP connections.

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

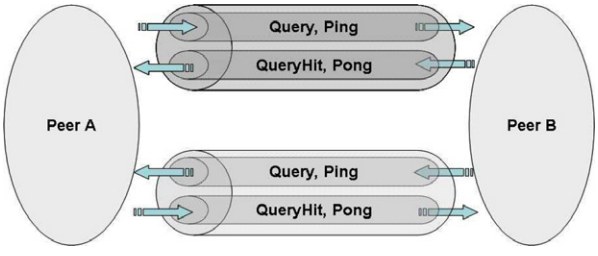

Fig. 4. Two OWRCs between two peers, which enable each peer to request service from the other.

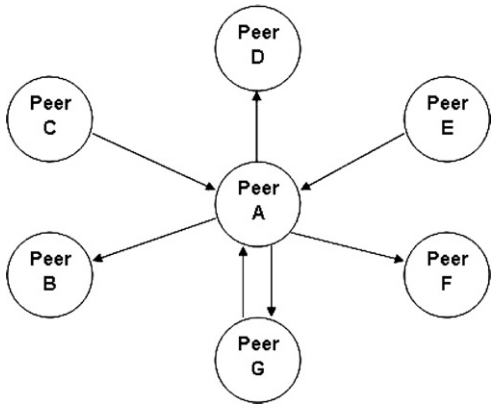

Fig. 5. A directed graph representation of a network consisting of OWRCs.

Fig. 5 shows an example model of a P2P network consisting of OWRCs. Here, peer A has 6 neighbors. It has four OUT-connected neighbors (B, D, F, G) and three IN-connected neighbors (C, E, G). In other words, the IN-connections of A are {C, E, G}, and the OUT-connections of A are {B, D, F, G}. When Peer A would like to search the network it can submit the Query only to its OUT-connected neighbors, namely B, D, F, and G. It will process the Queries only coming from its IN-connected neighbors (C, E, G). If it receives any Query from OUT-connected neighbors it drops the request. The details of a peer interaction with the PCMP are explained in Section 3.5.

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

3.3. Managing one-way-request connections

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

Each peer executing PCMP can maintain zero or more IN-connections, and zero or more OUT-connections. Maximum number of IN- and OUT-connections is limited by the available bandwidth and determined by peers. The following data structures can be used to define an IN and OUT connection.[2]

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

According to PCMP, connections are updated at a peer whenever that peer is involved in a download or upload operation; otherwise, PCMP does not update the connections of the peer.[3] The details of the PCMP operations that take place at requesting and providing peers are given below.

3.3.1. Managing IN-connections

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

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

- After the download, the downloader checks if there is an already created IN-connection coming from the uploader. If so, only the connection data structure is updated, i.e. the download counter is incremented by 1 and the last download time is set to the current time.
- If there is no existing IN-connection from the uploader to the downloader, a TCP connection is created between the downloader and the uploader.[4] The downloader waits for a Ping message from the uploader over the TCP connection. Because, after uploading, uploader is expected to request an IN-connection from downloader by sending a Ping message.

[2] Due to the power-law distribution of node degrees observed in P2P networks [4,34], we expect the average number of neighbors of a peer to be around 3–4, and therefore the overhead imposed by the solution on each peer will not be very large. This implies that the framework is scalable, thanks to its distributed nature.

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

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

- If the downloader receives the expected Ping message from the uploader, it proceeds with the following steps:
  - If the downloader can accommodate a new IN-connection, it creates a new connection to the uploader. It then replies with a Pong message to the uploader. In addition, it creates an IN-connection structure, setting the download counter to 1 and the last download time to the current time.
  - If there is no space to create a new IN-connection, connection replacement takes place. An existing IN-connection is replaced with the new IN-connection, i.e. the existing connection is released. The connection replacement policy is discussed in Section 3.4. Then, the downloader replies with a Pong message to the uploader. Again, the data structure for the connection is updated.

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

3.3.2. Managing OUT-connections

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

Algorithm 1. Sample pseudo-code for managing IN-connections. A peer X will execute this code after downloading a file from peer Y

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

The operations performed by the uploader to create an OUT-connection are described below.

- If there is an already-established OUT-connection at the peer to the downloader, the peer does not have to do anything, except possibly update some statistics.
- If there is no already-established OUT-connection to the downloader, the peer first creates a TCP connection to the downloader, through which further P2P messaging to create the OUT-connection can be done.[5] Then the uploader sends a Ping message to the downloader through this connection. Ping signifies that the uploader would like to establish an OWRC to the downloader. The downloader will consider the new OWRC an IN-connection, and it can either accept or reject the connection request. Normally, the downloader should accept the request if it obeys PCMP and if the downloaded file is not a fake file. The downloader will then send a Pong message back if it accepts the request.
- If a corresponding Pong message arrives from the downloader, the following operations are executed.
  - If the peer can accommodate a new OUT-connection, an OUT-connection to the downloader is created. The information about downloader is initialized: the downloader's ID is stored, Query Hit counter is set to zero, and the last Query Hit time is set to -1 (i.e. the value used when no Query Hit have been received yet).
  - If there is no space for a new OUT-connection, then the connection replacement policy is executed and one of the existing OUT-connections is replaced with the new connection.

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

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

Algorithm 2. Sample pseudo-code for managing OUT-connections. A peer Y will execute this code after uploading a file to peer X

```
Upload of a file F to a peer X has been finished;
OutConn = Search for an Out_Connection to Peer X;
if (OutConn is FOUND) then
  Update statistics;
else
  Send a Ping message to X;
  if (a Pong arrives from X) then
      newOutConn = Create_OUT_Connection();
      newOutConn.peerID = X;
```

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

```
        newOutConn.QueryHits = 0;
        newOutConn.LastQHitTime = − 1;
        if (there is space in the OUT_connection list) then
          Add(newOutConn, OUT_Connections);
        else
          victimOutConn = SelectVictim(OUT_Connections);
          Release(victimOutConn);
          Add(newOutConn, OUT_Connections);
        end if
      end if
    end if
```

3.4. Connection replacement policy

The connection replacement policy determines how to manage a limited number of IN and OUT-connections when all available connections of a peer are occupied and a new connection is required. There can be several different approaches for designing replacement policies. In this paper, we propose two connection replacement policies. In the first policy, the number of downloads or the number of hit messages provided from the neighboring peer is employed to decide which connection to replace. The connection with the least number of downloads or hit messages provided is selected as a victim. We call the PCMP protocol employing this policy *Contribution-based PCMP* (*C-PCMP*). In the second connection replacement policy, the time of the last download or the time of the last Query Hit provided from the neighboring peer is used to select the connection for replacement. The connection with the oldest time of the last download or hit messages provided is selected as a victim. We call the PCMP protocol that applies this policy *Time-based PCMP* (*T-PCMP*).

3.5. A Peer's actions and PCMP

3.5.1. Search

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

3.5.2. Forward queries

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

3.5.3. Forward Query Hits

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

3.5.4. Download

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

3.5.5. Upload

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

3.6. PCMP operation example

As a simple example, consider the P2P network topology given in Fig. 6. Assume each peer can only support up to 4 IN and 4 OUT-connections and the TTL is set to 2. The dashed circles represent the contributors (C1 and C2). In the given topology, the Query message of an indicated contributor (C1 or C2) cannot reach to the other one, since the indicated contributors are separated from each other by more than two hops. Assume a file F1 and a file F3 are stored on contributor C1, and a file F2 is stored on contributor C2. If the proposed PCMP is applied, the following scenario will occur.

- Peer P searches P2P network for file F1 with TTL 2. C1 replies with a Query Hit message. Then, Peer P downloads the file from the contributor peer C1. Upon download, Peer P deletes one of its IN-connections and adds a connection to C1 as a new IN-connection. C1 also removes (tears down) one of its OUT-connections and adds a connection to peer P as a new OUT-connection (see Fig. 7).

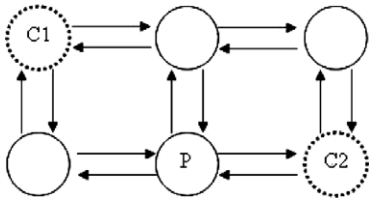

Fig. 6. A sample topology layout.

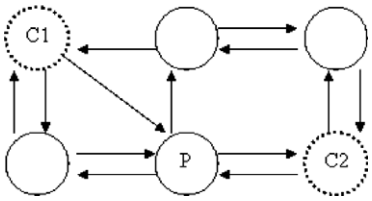

Fig. 7. After downloading, Peer P updates its IN-connection by adding C1.

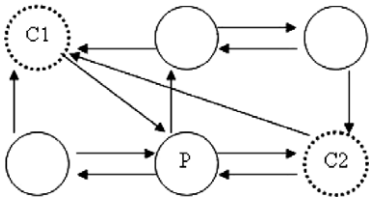

Fig. 8. After downloading, Peer C1 updates its IN-connection by adding C2.

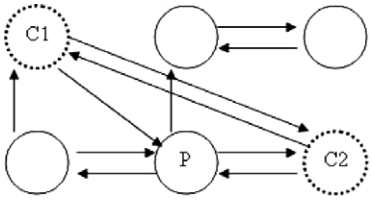

Fig. 9. After downloading, Peer C2 updates its IN-connection by adding C1.

- Later, contributor C1 searches for file F2 and the respective Query message reaches C2 via the peer P. C2 replies with a Query Hit message, and C1 downloads the file from C2. After downloading, a new connection is set up from C2 to C1. It is an OUT-connection for C2 and an IN-connection for C1 (see Fig. 8).
- Later, C2 searches for file F3, and C1 replies with a Hit message. After the download has been finished, a new connection is established between C1 and C2. This time the connection is established from C1 to C2; hence it is an OUT-connection for C1 and an IN-connection for C2 (see Fig. 9).

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

4. Performance evaluation

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

4.1. Overview of the simulation model

We used a simulation-based approach to study the model of a typical unstructured P2P network, namely Gnutella, with free riding and our PCMP incorporated. We implemented our simulation model including our PCMP protocol on the GnuSim P2P network simulation tool that we had developed earlier [13]. GnuSim was implemented as an event-driven simulator on the Windows platform using the CSIM 18 simulation library [9] and the C++ programming language. Interactions between peers and the P2P network, such as searching, downloading, pinging, etc., were implemented according to the Gnutella protocol specification given in [10].

Our model simulated a P2P network of 900 peer nodes. The peers were inter-connected to form a mesh topology at the beginning of a simulation run. For the base experiments with only the Gnutella protocol (i.e. without PCMP), we assumed that all the peers stayed connected in the same way until the end of the simulation runs.

We assumed that there were two types of peers in the simulated network: contributors and free riders. The properties of each peer type are summarized in Table 1. The properties of each peer type include the population ratio, shared file ratio, maximum number of simultaneous uploads possible, mean time between query generations, and whether peers replicate the downloaded files or not. The default values of each of these properties are set to values similar to those reported in [1,2,6,32,33].

There were 9000 distinct files, with four copies of each, distributed to the peer nodes at the beginning of each simulation run. These 36000 files were distributed among the peers and shared according to the file sharing ratios shown in Table 1. For the base experiments, we assumed that each file was of the same size and could be downloaded in 60 units of simulation time. In Section 4.3.5 we relax this assumption.

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

Each peer's upload capacity (the number of simultaneous uploads the peer could perform) was limited to 10. If a peer reached its upload capacity, any new upload

Table 1
Properties of peer types

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

requests were rejected. The querying peer could then try to download the file from another peer, selected from a list obtained from the Query Hit message. We assumed that the querying peer would repeat the same request a maximum of three times. After that, the peer would give up and could initiate a new search for another file.

We assumed that TTL is set to be 3 hops. In fact, Gnutella Protocol leaves TTL field value unsigned. In real life applications, TTL is usually set to 7. We set it to 3 in our simulation tests, since the network topology we simulate is small compared to the real world. If we had set TTL to 7, then most of the queries would have covered almost all of the peers, which would not have been realistic. In addition, we observed that changing the TTL value does not have an impact on the relative performance of Gnutella and our PCMP protocol.

Simulation experiments were run for 4000 units of simulated time. Each simulation was repeated 10 times and plotted on a 95% confidence interval.

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

4.2. Metrics

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

- *Total number of connections among contributors:* We count the number of connections (IN and OUT) which connect the contributors directly to each other. We expect that if the number of connections among contributors is increased, the contributors will get better service from the network. Since we assume the number of connections that a peer can have to be limited, those connections have to be used carefully by contributors. In order to get better service and more Query Hits, a contributor should have more connections to other contributors and less connections to free riders. In this way, a contributor can also reduce free riding through itself. This metric also shows how successful the PCMP protocol is in discovering and connecting contributors.
- *Total number of OUT-connections from free riders to contributors:* As stated in Section 3.2, if a peer has an OUT-connection to another peer, the peer can submit queries through this connection to that peer. Hence, the number of OUT-connections a peer has increases its chance to get replies and service from the network. Therefore, we count the total number of OUT-connections that free riders have towards contributors to measure how effective our protocol is in reducing free riders' access to resources.
- *Number of isolated free riders:* One of the aims of our protocol is to isolate free riders from contributors in the P2P network. If a free rider has no OUT-connection, then it cannot send any query and cannot receive any service, and we consider such a peer to be isolated. An isolated peer cannot download any files from the network. The greater the number of isolated free riders, the better it is for the network.

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

- *Number of downloaded files*: This is an important metric indicating the number of downloads that can be performed in a P2P network during a fixed time interval. If peers can download more files from the P2P network, then the level of satisfaction with the network will be higher.
- *Download cost*: We define the download cost for a peer as the ratio of the number of uploads to the number of downloads performed by the peer. This ratio indicates the load imposed on a peer compared to the service the peer gets from the network. The smaller this ratio is, the better it is from the perspective of the peer.
- *Number of P2P network protocol messages*: This metric shows the messaging overhead in the P2P network and the underlying infrastructure. Messaging overhead affects the scalability of a P2P system. The messaging overhead may be high due to the flooding approach used in querying, particularly in unstructured P2P networks. High numbers of protocol messages sent over the network also increase the level of congestion in the network.

4.3. Simulation results and analysis

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

4.3.1. Impact of PCMP on network topology

Fig. 10 shows the number of connections established among contributing peers over the simulation time. The results are for a P2P network employing our PCMP protocol using the time-based replacement policy (T-PCMP). As seen in the figure, the protocol causes more contributing peers to become directly connected to each other as time passes. By the end of the simulation time, the number of connections (IN and OUT) among contributors had increased from 309 to 562. Hence, connectivity among contributors increased by 82%.

Fig. 11 shows the number of OUT-connections of free riders to contributing peers plotted against the simulation

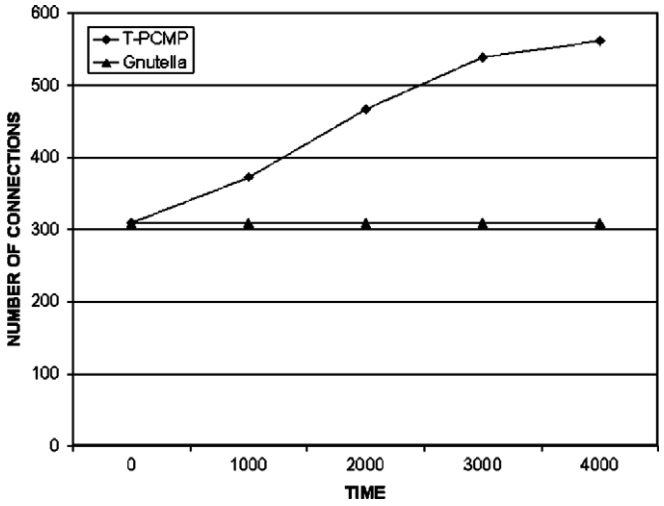

Fig. 10. Increase in the number of connections among contributing peers.

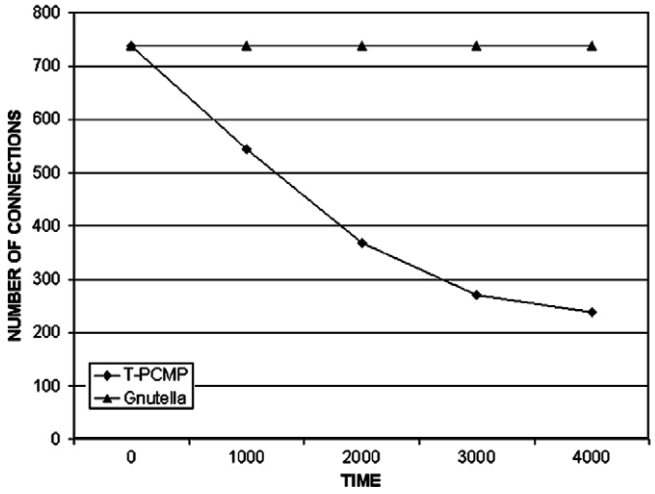

Fig. 11. Decrease in the number of OUT-connections from free riders to contributors.

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

Fig. 12 shows the number of isolated free riders over time. As time passed, more free riders were isolated from the network (they lost all their OUT-connections). At the end of the simulation time, a total of 24 free riders (out of 630) had been isolated.

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

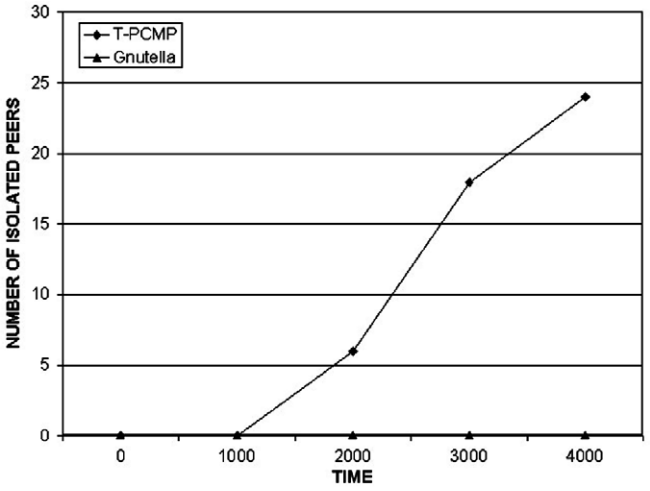

Fig. 12. The number of isolated free riders.

4.3.2. Impact of PCMP on P2P network performance

This section evaluates the effectiveness of our protocol in terms of the performance metrics described in Section 4.2.

4.3.2.1. Downloads of free riders. As Fig. 13 depicts, the number of downloads by free riders dropped when PCMP was applied. PCMP decreases OUT-connections of free riders towards contributors, and this reduces the chance of getting a hit on the queries. In this way, the number of downloads by free riders is reduced. Both C-PCMP and T-PCMP reduces the downloads. C-PCMP caused a 14% reduction, whereas T-PCMP achieved a 16% reduction.

4.3.2.2. Downloads of contributors. It is desirable to increase the number of downloads for contributors. Since each peer's upload capacity is limited, the download requests of contributors can sometimes be rejected. The rate of rejection is higher when there are many free riders in the system, so eliminating the effects of free riders on the P2P network will help to increase the number of downloads that contributors can make. This is indeed shown by Fig. 14;

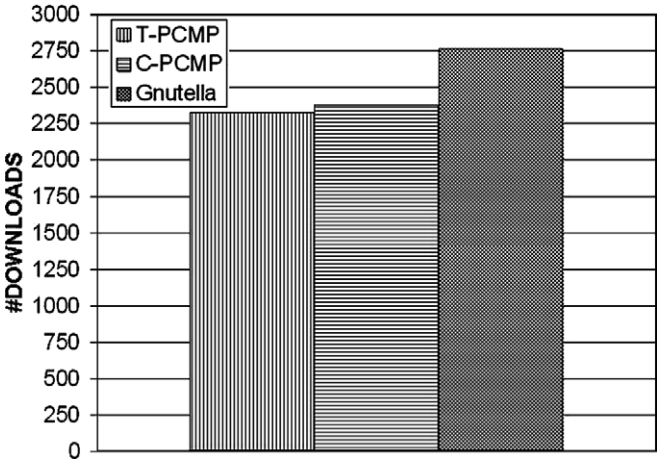

Fig. 13. Decrease in free riding peers' downloads.

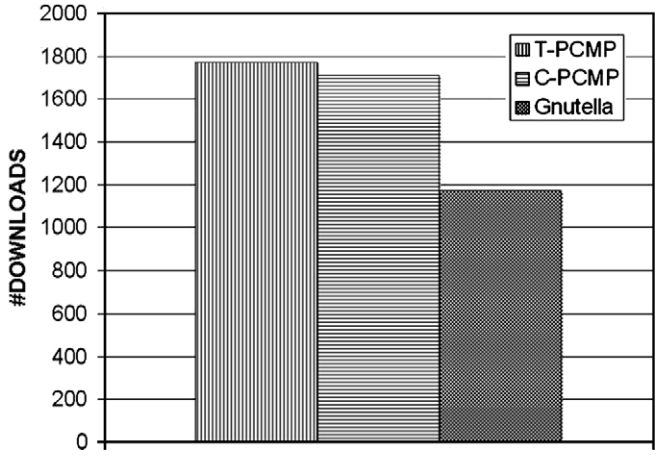

Fig. 14. Increase in contributors' downloads.

applying our PCMP methods achieved an increase in downloads done by contributors by 51%.

Fig. 14 shows that the improvement in downloads is slightly greater with T-PCMP than C-PCMP. While T-PCMP yielded an improvement of about 51%, the improvement when C-PCMP was used was about 46%.

4.3.2.3. Download cost. The load on a contributor can also be defined as the ratio of its uploads to its downloads. The results of our experiments show that our PCMP methods also cause a reduction in the download cost of contributors. As shown in Fig. 15, both T-PCMP and C-PCMP achieve a reduction of about 30% in the download cost for contributors.

4.3.2.4. Number of P2P protocol messages. The number of P2P protocol messages transmitted in the network is an important factor affecting scalability and bandwidth efficiency. PCMP results in a reduction of up to 36% in the number of transmitted P2P protocol messages (Query and Query Hit messages) originating from and destined for the free riders (Fig. 16). This result shows that applying the proposed PCMP helps a P2P network to handle more peers with less P2P messaging overhead and the system becomes more scalable with respect to the peer population. The reduction observed in the number of protocol messages is the result of reducing or stopping the propagation of Query messages from free riders. As the number of OUT-connections of free riders gets reduced, the propagation of Query and Query Hit messages for free riders will get reduced as well. The reduction of control traffic in a P2P network also means a reduction in the overhead imposed on the underlying infrastructure. This reduction translates to a better utilization of available bandwidths and to a decreased processing load on each peer.

4.3.3. Reactiveness of PCMP

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

To observe the reactiveness of PCMP, we conducted the following experiment. We randomly selected a probe node which initially behaved as a free rider. After a certain amount of time, the node changed its sharing attitude and began to share its files. We compared the level of service it got from the P2P network when it was behaving as a free rider and when it was sharing its files. The number of downloads that could be done by the probe peer is depicted in Fig. 17. As seen in the figure, when the peer begins to change its sharing attitude at a given time from free riding to contributing, PCMP reacts in a positive way and allows the peer to download more files.

4.3.4. Effects of peer and free rider population

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

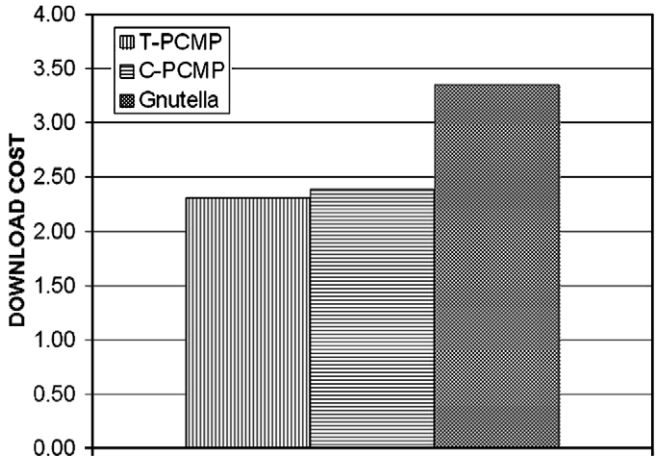

Fig. 15. Decrease in contributors' download cost.

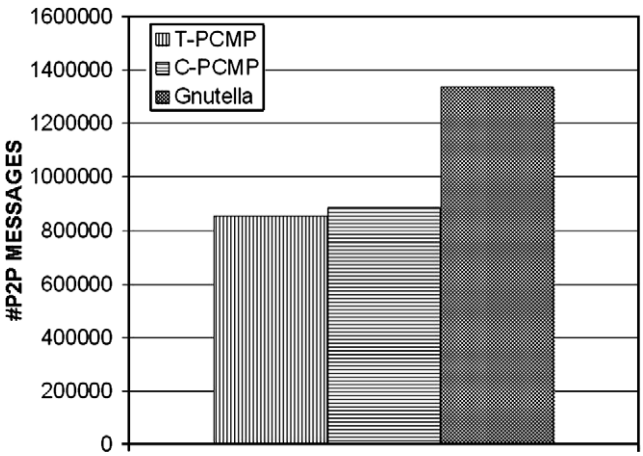

Fig. 16. Decrease in P2P messages from free riders.

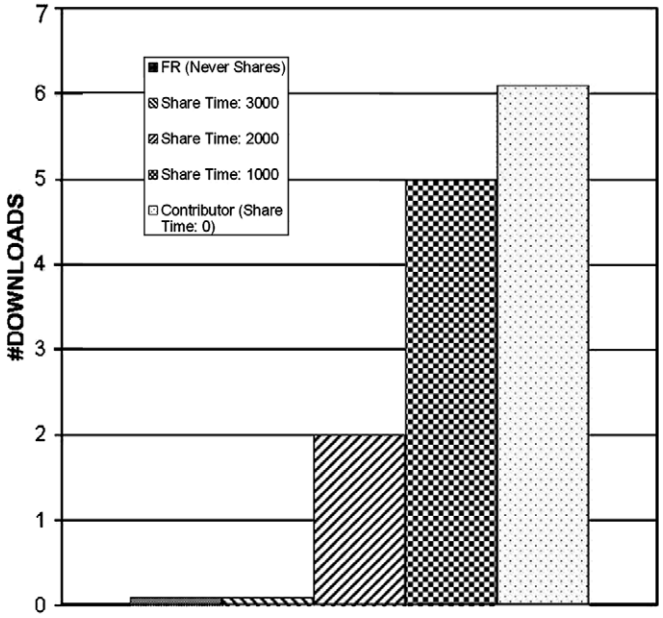

Fig. 17. Downloads of the probe node according to when it begins to share its files.

network's performance to be considerable. This is indeed what we have observed in the results of our experiments that were performed for various network sizes: 400, 900, 1600, 2500, and 4900 peers. Fig. 18 displays the performance in terms of the number of contributor downloads. As shown in the figure, the number of downloads by contributors is increased around 45% for all network sizes. Therefore, we conclude that increasing the number of peers in the network does not negatively affect the performance of our framework, and that our framework is scalable.

We also observed the effect of the size of the free rider population. As seen in Fig. 19, regardless of the ratio of free riders, T-PCMP achieves more downloads, around 50%, for contributors. Even at a low population ratio of free riders, the protocol performs very well.

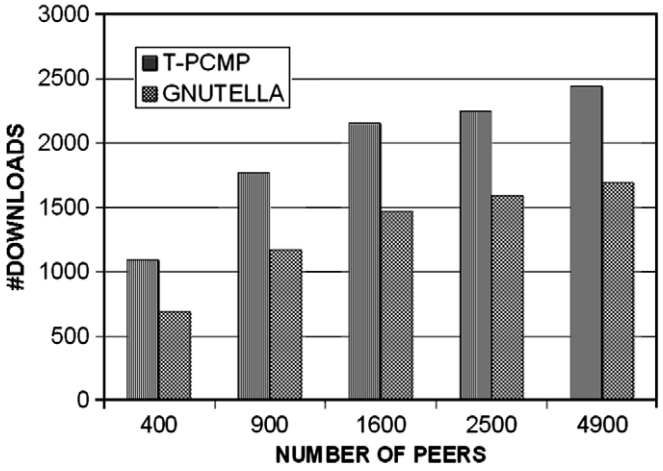

Fig. 18. The number of contributors' downloads when different numbers of peers are simulated.

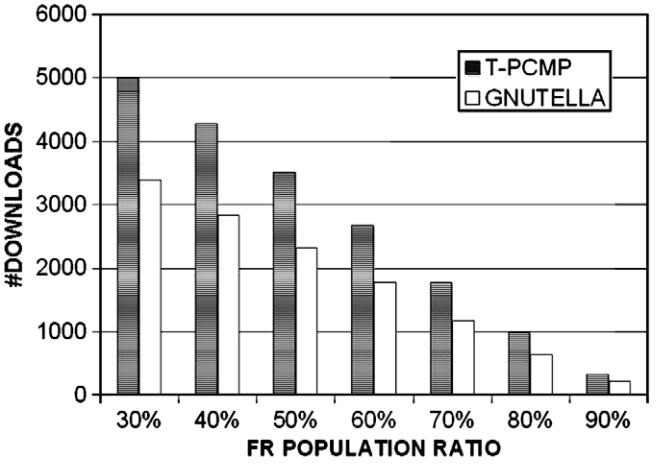

Fig. 19. The number of contributors' downloads when different free rider populations are simulated.

4.3.5. Effects of different file sizes and popularity

In Section 4.1, we assumed that each file is of the same size and the number of copies for each file is identical. In this section, we relax these assumptions by considering different file sizes as summarized in Table 2, and different levels of file replication as shown in Table 3. The values given in tables are based on the results of the P2P network observations done in [32,33].

We proposed two connection replacement policies in Section 3.4, namely T-PCMP and C-PCMP. To handle different file sizes we propose a new replacement method. In this method, the size of the file downloaded from the neighboring peer is used to select the connection for replacement. The connection with the least total amount of downloaded file is selected as a victim. We call the PCMP protocol that applies this policy *Size-based PCMP (S-PCMP)*.

Fig. 20 shows the results of different file sizes on the contributor downloads. PCMP increases the contributor downloads as much as 55% compared to Gnutella.

Table 2
Properties of different file sizes

File type	File size	Ratio (%)
Very small	~ 0.3	10
Small	~5 MB	50
Medium	~40 MB	20
Large	~100 MB	10
Very large	>100MB	10

Table 3
Properties of different levels of file replication

Name	Group A (ratio/replication)	Group B (ratio/replication)
Rare	10% of files: 1 copy	90% of files: 4 copies
Popular	10% of files: 40 copies	90% of files: 4 copies
Uniform	All files: 4 copies	All files: 4 copies

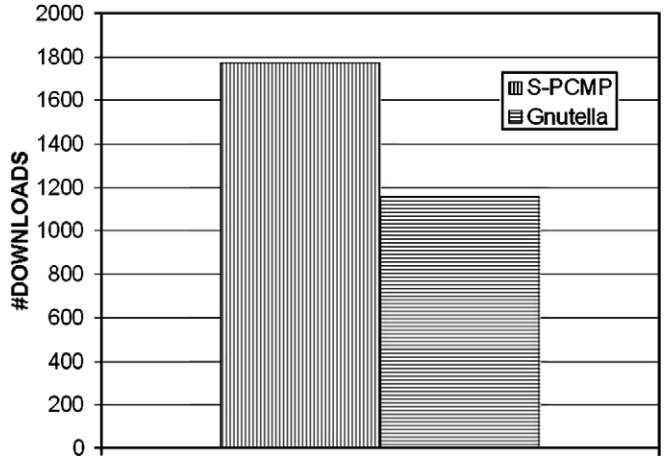

Fig. 20. The number of contributors' downloads with the existence of different file sizes.

For evaluating the impact of different file replication levels, we used three file replication schemes as summarized in Table 3. We split the files into two groups and replicated them with different factors. In the RARE distribution, 10% of the files are rare (fewer replications) compared to 90% of the files. Similarly, in the POPULAR distribution, 10% of the files are more popular (more replications) than those of 90% of the files. In UNIFORM (default) distribution all the files have the same number of copies.

The results of the simulation tests are depicted in Fig. 21. The figure summarizes the effects of different file distribution schemes on the contributors downloads. With all the file distributions considered, PCMP performs about 55% better than Gnutella. Total number of downloads of contributors is affected by the distribution strategy of file copies. However, PCMP manages to profit the contributors with all different types of file distribution schemes evaluated.

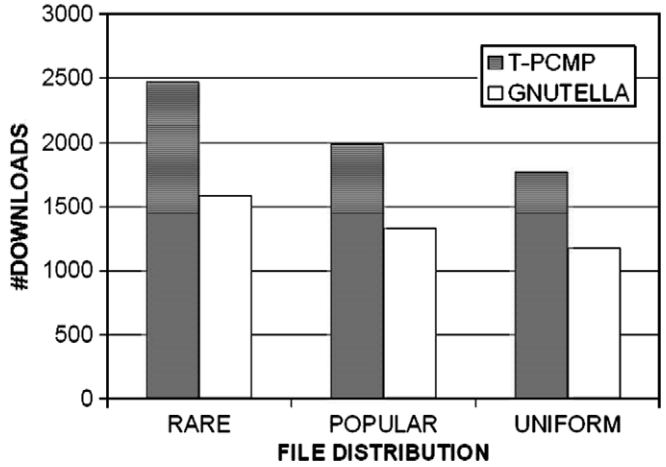

Fig. 21. The number of contributors' downloads when different file replications exist.

5. Possible attacks

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

5.1. A malicious peer does not comply with the proposed PCMP rules

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

If all free riders apply this attack, then contributors establish OUT-connections only with other contributors, and this automatically helps them to become more connected with each other. In the end, contributing peers will have an advantage over free riders, since a peer has a restricted number of OUT-connections and a contributor will not waste them for connections to free riders. Because, as discussed in Section 3.2, if a contributor uploads a file to a peer, the contributor will update its OUT-connection with that peer. If there is no free OUT-connection, then it will drop an existing OUT-connection and add the new peer. If the dropped connection is with a contributor and the newly added connection is with a free rider, the contributor will not benefit from the new connection since free riders do not share almost any files. However, the contributors are not aware if a peer is a free rider or not. If free riders reject IN-connection requests by not sending a Pong message, then the contributors will not update their OUT-connections. The contributors will only update their OUT-connections when they upload files to other contributors, since other contributors will accept the IN-connection requests by replying with Pong messages. Therefore, we expect that this attack will not affect the contributors much.

In order to observe the effects of this possible attack, we designed a new simulation setting. In the new simulation, we assumed that all free riders would reject creating an IN-Connection from a source peer after downloading a file. As seen in Fig. 22, this attack does not adversely affect the download performance of the contributors as compared to the results given in Fig. 14. On the contrary, the contributors can download slightly more files, because they become more closely connected to each other, as seen in Fig. 23.

5.2. A malicious peer replies with a faked Query Hit

To establish OUT-connections, a malicious peer can reply to a Query message as if it has the file. However,

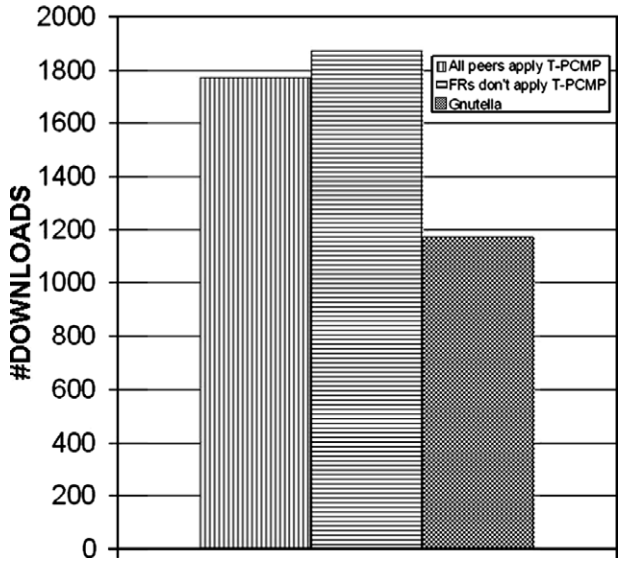

Fig. 22. The number of contributors' downloads when free riders are noncooperative.

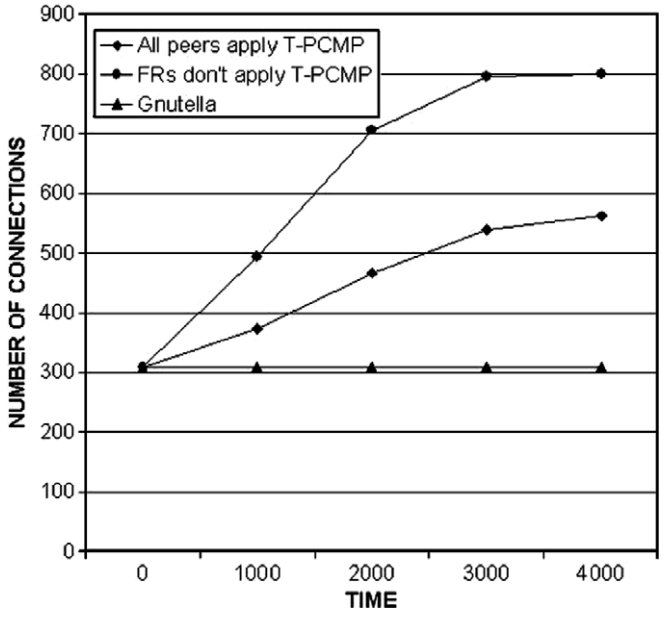

Fig. 23. Increase in the number of connections among contributing peers when free riders are noncooperative.

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

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

To increase the number of OUT-connections, a malicious peer can request OUT-connections from peers as if it is a new peer in the network. If the peers accept all newcomers' connection requests without any limitations, the attacker can benefit from this situation. Jakobsson and Juels proposed a method of combating such problems: proof of work (POW) protocols [15]. The main idea of these protocols is that *a prover* demonstrates to *a verifier* that it has expended a certain level of computational effort in a specified interval of time. POWs were proposed as a mechanism for a number of security goals including server access metering, construction of digital time capsules, uncheatable benchmarks, and protection against spamming and other denial of service attacks. However, in [31], it was argued that the implementation of POW to decrease spamming to very low levels could limit small number of legitimate user's activities as a side effect. In our context, we believe that we can implement POW as an effective discouraging method against Free Riders. There are no side effects similar to the ones mentioned above in our application. In our work, we can implement POW to minimize these attacks to very low levels. Thus creating new connections can cost time, limiting the ability of the attackers to request them without a limit. We can include a rule in the general P2P protocol for initial connections stating that clients are required to solve a puzzle, such as factoring a number, before a Ping request is answered with a Pong message. The puzzles could require additional work as resources become more scarce. This increases the resources required by attackers to attack the system proportional to the threat of the attack.

6. Conclusion

In this paper, we propose a novel approach and a connection management protocol (PCMP) against free riding in unstructured P2P networks. Our approach is based on dynamically adapting P2P network topology via our PCMP protocol to promote contribution in the network. The PCMP protocol manages the connections among peers based on the amount of contributions by peers. PCMP is simple to implement, has low overhead to run, fully complies with the concepts and protocols of unstructured P2P networks, and is decentralized so as to operate efficiently.

By adapting the overlay topology, we aim to reduce the amount of free riding and its adverse impact on P2P networks, and 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 peers, and the scalability of the network. As the performance results of simulation experiments indicate, the protocol does indeed reduce the adverse effects of free riding on a P2P network, and the performance of the P2P network is improved considerably.

It is possible to conceive of various attacks and workarounds that free riders can try to bypass the protocol. However, we show that our solution can cope with possible attacks. Furthermore, simulation experiments prove that most of the possible attacks do not render our protocol ineffective.

References

[1] Eytan Adar, 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, in: IEEE International Symposium on Cluster Computing and the Grid, May 2002, 65–74.

[3] Lakshmish Ramaswamy, Ling Liu, Free riding: a new challenge to Peer-to-Peer file sharing systems, in: Annual Hawaii International Conference on System Sciences – Track7, Big Island, Hawaii, January, 2003.

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

[5] Matei Ripeanu, Ian Foster, 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 6 (1) (2002).

[6] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble, A measurement study of Peer-to-Peer file sharing systems, Multimedia Computing and Networking (2002).

[7] Ramayya Krishnan, Michael D. Smith, Zhulei Tang, Rahul Telang, The impact of free-riding on Peer-to-Peer networks, in: Annual Hawaii International Conference on System Sciences – Track 7, 2004.

[8] Vivek Vishnumurthy, Sangeeth Chandrakumar, Emin Gun Sirer, KARMA: a secure economic framework for P2P resource sharing, in: Workshop on the Economics of Peer-to-Peer Systems, 2003.

[9] Herb Schwetman, CSIM: A C-based, Process Oriented Simulation Language, in: Winter Simulation Conference, 1991.

[10] Clip2, The Gnutella Protocol Specification v0.4 (Document Revision 1.2), "http://www9.limewire.com/developer/gnutellaprotocol0.4.pdf", 2001.

[11] M. Karakaya, I. Korpeoglu, O. Ulusoy, A distributed and measurement-based framework against free riding in Peer-to-Peer networks, IEEE International Conference on Peer-to-Peer Computing (2004).

[12] Nazareno Andrade, Francisco Brasileiro, Walfredo Cirne, Miranda Mowbray, Discouraging free-riding in a Peer-to-Peer grid, in: IEEE International Symposium on High-Performance Distributed Computing, 2004.

[13] M. Karakaya, I. Korpeoglu, O. Ulusoy, GnuSim: a Gnutella network simulator, in: 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".

[14] Qixiang Sun, Hector Garcia-Molina, SLIC: A selfish link-based incentive mechanism for unstructured Peer-to-Peer networks, in: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), 2004.

[15] M. Jakobsson, A. Juels, Proofs of work and breadpudding protocols, in: Proceedings of the Communications and Multimedia Security, September 1999.

[16] Y. Liu, Z. Zhuang, L. Xiao, L. Ni, AOTO: adaptive overlay topology optimization in unstructured P2P systems, The IEEE Global Telecommunications Conference (Globecom) (2003).

[17] Curt Cramer, Kendy Kutzner, Thomas Fuhrmann, Bootstrapping locality-aware P2P networks, in: The IEEE International Conference on Networks (ICON), 2004.

[18] A. Singh, M. Haahr, Topology adaptation in P2P networks using Schelling's model, in: The Workshop on Emergent Behaviour and Distributed Computing, PPSN-VIII, 2004.

[19] D. Hughes, G. Coulson, J. Walkerdine, Free riding on Gnutella revisited: the bell tolls? IEEE Distributed Systems Online 6 (6) (2005).

[20] M. Yang, Z. Zhang, X. Li, Y. Dai, An empirical study of free-riding behavior in the maze P2P file-sharing system, IPTPS, 2005.

[21] Bram Cohen, Incentives build robustness in bittorrent", Workshop on Economics of Peer-to-Peer Systems, vol. 6, 2003.

[22] eDonkey Web Site, "http://www.edonkey2000.com", 2006.

[23] eMule Web Site, "http://www.emule-project.net", 2006.

[24] MyungJoo Ham, Gul Agha, ARA: a robust audit to prevent free-riding in P2P networks, in: The Fifth IEEE International Conference on Peer-to-Peer Computing (P2P2005), 2005.

[25] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The EigenTrust algorithm for reputation management in P2P networks, in: The 12th International Conference on World Wide Web (WWW), 2003.

[26] B. Yang, H. Garcia-Molina, PPay: micropayments for peer-to-peer systems, in: Proceedings of the 10th CCS, V. Atluri and P. Liu (Eds.), ACM Press, New York, 2003, pp. 300–310.

[27] Prashant Dewan, Partha Dasgupta, Securing P2P networks using peer reputations: is there a silver bullet? IEEE Consumer Communications and Networking Conference (CCNC 2005) (2005).

[28] Dipyaman Banerjee, Sabyasachi Saha, Sandip Sen, Prithviraj Dasgupta, Reciprocal resource sharing in P2P environments, in: The 4th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05), July, 2005.

[29] H. Cai, J. Wang, Foreseer: a novel, locality-aware Peer-to-Peer system architecture for keyword searches, in: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, 2004, pp. 38–58.

[30] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Scott Shenker, Making Gnutella-like P2P systems scalable, in: Proceedings of ACM SIGCOMM, 2003.

[31] B. Laurie, R. Clayton, Proof-of-work proves not to work, in: The Third Annual Workshop on Economics and Information Security, 2004.

[32] N. Leibowitz, A. Bergman, R. Ben-Shaul, A. Shavit, Are file swapping networks cacheable? Characterizing P2P traffic, in: Proceedings of the 7th Int. WWW Caching Workshop, 2002.

[33] N. Leibowitz, M. Ripeanu, A. Wierzbicki, Deconstructing the Kazaa network, in: The Third IEEE Workshop on Internet Applications, WIAPP 2003.

[34] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer networks, in: ICS'02, New York, USA, June 2002.

Murat Karakaya is currently a Ph.D. candidate in the Computer Engineering Department of Bilkent University in Ankara, Turkey. His current research interests include peer-to-peer networks and mobile database systems.

İbrahim Körpeoğlu received his Ph.D. and M.S. degrees from University of Maryland at College Park, both in Computer Science. He is currently an Assistant Professor in the Computer Engineering Department of Bilkent University, Ankara, Turkey. Prior to joining Bilkent University, he worked in Ericsson, IBM T.J. Watson Research Center, Bell Labs, and Telcordia Technologies, in USA. His research interests include computer networks, wireless ad hoc and sensor networks, mobile computing, and P2P networks.

Özgür Ulusoy received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. He is currently a Professor in the Computer Engineering Department of Bilkent University in Ankara, Turkey. His current research interests include peer-to-peer and mobile systems, web querying, and multimedia database systems. He has published over 80 articles in archived journals and conference proceedings.