A Market-Managed Topology Formation Algorithm for Peer-to-Peer File Sharing Networks

Tarik Idris[٤], Jörn Altmann[٤٩]

[٤] School of Information Technology, International University in Germany
76646 Bruchsal, Germany
[٩] Techno-Economics & Policy Program, College of Engineering, Seoul National University
Seoul 151-744, South Korea
jorn.altmann@acm.org, tarik.idris@i-u.de

Abstract. Currently, peer-to-peer (P2P) networks suffer from users that do not contribute any kind of resources to the P2P community. Those users, which are called freeriders, benefit largely from contributions of other users but reduce the system performance for contributing users. This paper proposes an incentive scheme for P2P networks that motivates users to collaborate within the system. The solution that we propose has an impact on the topology formation of a P2P network. Using our market-managed topology formation algorithm (IUTopForm) for P2P networks, contributing users will be clustered within clubs that are different to clubs of freeriders. The differentiation is possible because of a reputation system, which considers users' past contributions. The effect of this approach is that service requests of freeriders will take longer to be answered (if at all) than service requests of resource-contributing users. We illustrate this effect through measurements with our P2P network simulator. We also show that clubs are only interconnected if the difference in their reputation values is not large. The comparison with Bagla and Kapalia's approach, which inspired our work, shows that the IUTopForm approach improves the overall utility of the system. The utility function and the topology formation algorithm are described in detail within this paper.

Keywords: Incentive Scheme, Pricing, Peer-to-Peer Networks, Simulation, Market-Management, Trust, Reputation, Economics, Utility Function.

1 Introduction

Although there have always been applications that used the peer-to-peer (P2P) paradigm (e.g. *USENET* [22] and *FidoNet* [12]) from the beginning of the network era, this technology has received much more attention during the last 5 years. On the one hand, this was caused by widespread availability of network computers and decreasing bandwidth costs. On the other hand, "killer applications" like *Napster*, *Kazaa*, and *Gnutella* were developed, which allow file sharing in a very simple way. However, P2P networks suffer from two issues. First, in order to stop illegal file sharing, the music industry injects faked files into the network. Second, and even more severe, many P2P network users do not provide any resources to the P2P network. These so-called freerides benefit highly from contributions of other users. Consequently, resource-contributing/cooperating users suffer in such a network from reduced performance.

In order to solve this problem, an incentive scheme has to be integrated into P2P networks. An incentive scheme provides benefits to those users who contribute resources (may it be content or hardware) to the network and reduces the performance of users who do not contribute. A few existing approaches on incentive schemes have been proposed during the last years. The one that we consider in this paper are from Asvanund, Bagla, Kapadia, Krishnan, Smith, &Telang [5], Walsh & Sirer [23], Yang, Chen, Zhao, Dai, & Zhang [26], Cohen [8], and Feldman, Papadimitriou, Chuang, & Stoica [10]. While the first four describe implementations of incentive schemes, the last analyzes the importance of incentive schemes for the success of P2P networks.
Our approach, the market-managed topology formation algorithm (IUTopForm) is based on a new utility function, which considers the amount of shared content of a user, the distance to another user, the similarity in taste with another user, and the other user's reputation. Since we especially focus on P2P file sharing networks, the content that the utility function considers are files. Note, any future reference to a P2P network will be in the context of a file sharing application. The topology formation algorithm, which we present in this paper, achieves significant performance improvements for P2P networks through a combination of different approaches. These approaches are:

- The definition of datasets to find similar interests
- A use of a reputation system
- The definition of a utility function for users
- The definition of a utility function for clusters (clubs)

The remainder of the paper is organized as follows. Section 2 gives an overview about the problems that P2P file sharing networks faced in the past. The architecture of version 0.6 of the Gnutella network, the version that our simulation is based on, is described in Section 3. An overview about previous research on incentive systems for P2P systems is given in Section 4. Our topology formation algorithm is described in Section 5 and Section 6. Finally, we conclude by presenting our measurement results in Section 7.

2 File Sharing of Music Files and Countermeasures

Napster was first released in the fall of 1999 and became increasingly popular during 2000, introducing millions of users to P2P file sharing, more specifically, to sharing of digital music files in MP3 format. Napster's architecture was not purely distributed, since the search for files was performed on a central server. Therefore, in July 2001, the service could easily be shut down by a judge's order after the music industry successfully sued Napster for copyright infringement [16].

Given the success of Napster, it came as no surprise that many purely decentralized file sharing applications were created to provide a similar service to about 13 million users that Napster had during its peak. One of the earliest attempts was Gnutella, in March 2000. The first client was developed by *Nullsoft*, a small developer studio owned by *AOL*. Its development continued in several distinct projects. Nowadays, there are numerous different clients interoperating by the standards defined by the Gnutella Developer Forum (GDF) [23]. The original Gnutella protocol is nowadays referred to as Gnutella version 0.4 and is rather outdated, although it was a technical revolution at its time. Changes and extensions, which are incorporated in all major clients, will soon be standardized as Gnutella version 0.6 by the GDF [13]. However, scalability issues with the 0.4 protocol hindered a success similar to Napster's.

The *FastTrack* network and its most popular client Kazaa did not have these technical problems. The FastTrack protocol is fully decentralized and allows for swarming, i.e. downloading different parts of the same file from several hosts to increase throughput through parallelization.

Recently there has been a significant decline in users on the FastTrack network [17]. This is attributed to several factors. Firstly, the lack of innovations in the P2P clients, the protocol, and

the bundled *adware* and *spyware* that has plagued users of the official clients, has driven the more tech-savvy users to alternative networks like Gnutella and *eDonkey*. Secondly, the lawsuits by the music industry have specifically targeted Kazaa hosts sharing over 1000 files, causing users to reduce the amount of shared content or to completely desert the network. Last, but not least, the music industry's tactic to introduce bogus files into the network, sometimes described as "polluting the pool", has had success in spoiling the file sharing experience. These files are offered by hosts under the control of firms like *Overpeer*, which specialize in anti-piracy solutions. The files contain either looped parts of the advertised song/movie or plain noise. Because of a design flaw in the FastTrack protocol, only parts of a file are hashed, allowing malicious nodes to advertise bogus files with the same hash as the respective original file. Because the clients download segments of a file from multiple sources, only one source has to be malicious to corrupt the downloaded file. Another problem is the increased efficiency of these firms in mimicking the appearance and behavior of "normal" file sharers [1].

It can be expected that the arms race between the P2P developers and the content distributors will continue in the near future. The latest releases of Gnutella and eDonkey clients block certain IP-ranges containing malicious nodes. As a next step, copyright holders might release modified versions of open-source clients that interfere with normal network operation. A possible answer for the Gnutella vendors would be to only accept connections to trusted clients, i.e. clients with a high reputation value.

3 The Gnutella Protocol

For this work, the Gnutella protocol has been chosen, since it is widely used and researched. It has been developed in an open-source process, making information about it easily available. Nevertheless, the algorithm that has been developed within this work could be applied to any other P2P network using the *Ultrapeer* paradigm and relying on flooding for resource discovery (e.g. the FastTrack protocol).

The stable version of the Gnutella protocol is version 0.4, but this version is no longer in use in its pure, original form, since it has been shown that it does not scale up to bigger network sizes [19]. Due to the open nature of the protocol, with several independent clients using it, there is no strict standard adhered to by everybody. However, the basic features defined in the specifications for version 0.6 are widely adopted and will be followed in this work.

3.1 Basic Architecture of the Gnutella Network

The Gnutella network, often referred to as the *GNet*, is fully decentralized, i.e. there are no central servers. There are two types of nodes in the GNet, Ultrapeers and *Leafs* (see section 2.3 of the protocol definition in [18]). A Leaf is connected to several Ultrapeers (usually three) and does not have connections to other Leafs (Fig. 1). An Ultrapeer is a reliable, powerful node that handles most of the routing, so that the majority of nodes (Leafs) is not overwhelmed by the overhead of network organization. The number of Leafs that an Ultrapeer is connected to depends on the user's choice and the specific client used, and usually lies between 30 and 300 Leafs. Ultrapeers also maintain connections to several (usually five) other Ultrapeers (Fig. 1).

Ultrapeers use the *Query Routing Protocol (QRP)* to route requests for files. A file request is only forwarded to a Leaf that has been determined to be able to answer the file request. The QRP is based on hashing filenames of shared files, accumulating the hashes in a table (QRP table) and

1 In the beginning, it was rather simple to identify malicious users and bogus files. User names were generated by a simple scheme (commonly used words concatenated by a two digit number) and the file names contained phrases like "no loops" or "real version". The host that offered the bogus file always had broadband connections. Now, all these give-away-clues have been eliminated.

storing the tables at the Ultrapeer [20]. Therefore, an Ultrapeer can be seen as an indexing server for the connected leafs.

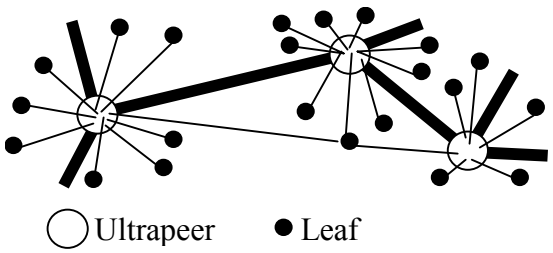

Fig. 1 The two types of nodes in the GNet are Ultrapeers and Leafs

The election of Ultrapeers is self-organized (see section 3.7 of the specifications [18]). There are several requirements that a node has to fulfill to be considered capable of becoming an Ultrapeer. The requirements state that it must not be located behind a firewall, have sufficient resources (CPU, RAM, and bandwidth), and a high uptime. An Ultrapeer-capable node can become an Ultrapeer, if there is a need for more Ultrapeers, i.e. there are only a few Ultrapeers with open connection slots.

3.2 Finding an Entry Point into the Network

To connect to the GNet, a node must know the IP address of at least one other connected node. This is done by accessing node addresses stored during a previous successful run or by referring to a GWebCache. This is a script, which is located on a Web server and stores node addresses and URLs of other GWebCaches. If there are no node IP addresses and no working GWebCaches known to a node, it will not be able to make a connection to the Gnet (see sections 2.1 (Bootstrapping) and 3.2 (The Web Caching System) [18]).

3.3 Querying the Network

As described in section 2.7 [18], a node sends Query packets to connected nodes to find out who offers files with filenames containing certain search terms in the GNet. A Query packet contains a 16-byte long *Global Unique Identifier (GUID)*, a *Time-To-Live (TTL)*, and a *hops* value. Before a node forwards a Query packet, the TTL is decremented and the hops are incremented. If the TTL becomes zero, the Query will be dropped. Otherwise, it will be sent to the connected Ultrapeers and to the connected Leafs if the QRP demands it. A Query is never forwarded to the node that sent it. If a Query with the same search term and the same GUID has already been received, the duplicate is dropped. Normal values for the TTL are three or four, because this seems to be a good compromise between coverage and generation of network load.

If a Query is received by a node and the search terms match one or more files, a *QueryHit* is generated. The QueryHit contains the issuing node's IP address, matching filenames, the same GUID as the Query, a TTL that is one higher than the number of hops the Query took to reach the answering node and a hops value of initially zero. A QueryHit is treated similar to a Query, with the difference that it is only forwarded along the way the associated Query came. For this purpose, each node keeps a routing table, containing information about received Queries. Once a querying node receives a valid answer, it will negotiate the file transfer directly with the responding node determined by the IP address in the QueryHit packet.

4 Existing Incentive Schemes for P2P Networks

The goal of incentive schemes is two-fold: Firstly, the sharing of files should be encouraged; secondly, traffic costs are supposed to be reduced. We describe three of these schemes in this section. However, only the scheme that is the basis for our scheme is explained in more detail.

4.1 Credence P2P Network

Walsh & Sirer developed an incentive scheme for their P2P network in order to exclude malicious users [23]. The incentive scheme is based on a voting system. If a node needs to make a decision whether it can trust another node, it requests votes about the other node from the network. Based on the returned votes, the weighted average of votes is calculated. The weights are set according to experiences from earlier requests. This algorithm is based on a reputation system. It does not consider any kind of topology.

4.2 Maze P2P Network

Yang, Chen, Zhao, Dai, & Zhang developed a P2P file sharing application (called Maze) with a centralized search engine [26]. One of Maze' main characteristics is its evolving incentive scheme. It is based on a set of incentive policies that are driven by user feedback from forums. Users get points awarded for uploading files and get points subtracted for downloads.

In detail, a new node gets 4096 points on its account P initially. Each Mbyte uploaded results in 1.5 points. The cost of a download is tiered, depending on the byte volume: 1 point per Mbyte within the first 100 Mbyte; 0.7 points per Mbyte between 100 and 400 Mbyte; 0.4 points per Mbyte between 400 and 800 Mbyte; 0.1 points for each Mbyte thereafter. Users with an account of less than 512 points get at most a bandwidth capacity of 300Kb/s. Otherwise, each request is ordered according to the value $T = \text{requestTime} - 3 \log (P)$. Requests with a top ranking get more resources. However, this has not been specified in the paper.

Besides, it has to be noted that the success of this scheme is influenced by the fact that users with a high value on their account get positive recognition (i.e. prestige, respect) within the online forum. The shortcoming of this incentive scheme is that it does not evaluate node/user behavior (e.g. offering mislabeled or corrupt files), it simply measures all file transfers.

4.3 The Club Approach

As introduced by Asvanund et al. [5], the club approach uses economic measures to create a more efficient network overlay topology for the Gnutella network. To reach these goals, they introduced the economic concept of a club. Clubs consist of one Ultrapeer and its connected Leafs and are described as "content-based, self-organizing communities of peers".

A node selects a club depending on the utility that it gets when joining the club. Each node tries to maximize its utility by attempting to join clubs, which would provide the highest utility to them. To determine the net utility that a club delivers, the sum of the costs incurred by each club member is subtracted from the sum of all the utility added by each club member. The utility and the cost term are scaled by two node-dependant factors. A club only accepts a node's connection request if this will lead to a higher club utility. Club utility is defined as the sum of utilities of each connected node in this club.

The utility that a node *y* provides to a node *x* has been defined as the similarity *sim* of the content of *y* to the past queries of *x*, multiplied by the weighted amount of content *y* shares and multiplied by the sum of the bandwidth of *y* and the weighted distance between *x* and *y*. The similarity function (sim) is based on information retrieval methods analyzing the filenames of the shared content. The distance and bandwidth are also scaled by two node-dependant factors. The

distance term is a function that assigns high values to nodes located close-by in the underlying physical network topology.

The costs that a node *y* imposes on a node *x* are defined as the similarity of the past queries of *y* to the content of *x*, multiplied by the amount of content shared by node *x*, multiplied by the sum of bandwidth minus the distance between *x* and *y*. The similarity and distance functions are the same as used in the value function. This formula assigns higher cost to Leaf nodes whose information needs have a high probability of being satisfied, have high bandwidth, and are located on a different backbone.

Although, as described in [5][6], this approach yields a better network performance for nodes sharing content than the standard Gnutella protocol in version 0.6 (and version 0.4), this algorithm does not generate an optimal P2P topology. The sub-optimality is caused by the fact that no specific utility function has been defined for Ultrapeers. It does not specify how inter-club connections are evaluated nor made. The utility function itself does not differentiate between successful queries (i.e. queries that returned results) and unsuccessful queries (i.e. queries that did not return results). Since unsuccessful queries are much more likely to be reissued than successful queries, a club that can answer the former is more useful than one that can answer the latter.

5 Topology Formation Algorithm: IUTopForm

Our algorithm is based on the club approach described in the previous section. However, it differs significantly in five items, namely, the system for predicting future information demands; the reputation system; the utility function for Leaf nodes; the utility function for interclub connections; and the consideration of bandwidth.

5.1 Prediction of Future Information Needs

The first characteristic of our algorithm is that it uses two different ways to predict future information needs. First, the algorithm considers the unsuccessful queries issued by a node and compares them to the other node's shared content. The underlying assumption is that a user will repeat queries that had no results, since a user's past file needs that could not be fulfilled remain relevant in the future. Second, the algorithm predicts future informational needs by comparing a node's shared content to the other node's shared content. The underlying assumption here is that users have a certain "taste" in files, i.e. their future informational needs can be predicted by their past informational needs. For example, users that only listen to certain genres of music (e.g. classical music, movie soundtracks, etc) will have a certain number of identical files. Therefore, the users who share the same taste are more likely to answer each other's queries.

There is a limit to the needed similarity in content, though. If two users have the exact same set of files, they cannot satisfy each other's informational needs. All files of one node are already in the other's collection. Therefore, a target similarity that a node wants to achieve can be specified as well. The best value needs to be determined empirically in future research.

5.2 The Reputation System

With the recent success of P2P networks, the number of users who are selfish (e.g. freeriders, who adhere to the rules in a way that benefits them but hurts other users) or who are malicious (e.g. hackers or firms like Overpeer Inc. who intend to disrupt the network operations) has increased significantly. This behavior of users could be captured through a reputation system, like the ones described by Kung & Wu [14], Dutta, Goel, Govindan & Zhang [8], Aberer & Despotovic [3], Abrams, McGrew & Plotkin [4], and Lee & Hwang [10]. Our topology

formation algorithm assumes a reputation system that assigns higher reputation values to cooperative (i.e. not selfish and not malicious) nodes. Those nodes earn reputation points by providing services to other nodes. How reputation is gained, where it is stored, and how it is verified is transparent to the topology formation algorithm introduced here. A node's reputation is an element of the utility function and is therefore taken into account for each utility calculation. Trustworthy nodes will more likely encounter other trustworthy nodes in a club they are part of than untrustworthy nodes. This also leads to the aggregation of untrustworthy nodes in their own clubs, minimizing their negative impact on the network.

5.3 Utility Function for Leaf Nodes

Our algorithm uses the following utility functions for making decisions about joining or leaving a club:

$$\begin{aligned} \mathrm{LeafU}_x(\mathrm{club}) &= \Sigma_{y\ ,\ \mathrm{club}}\, U(x, y) \\ &\quad + a_1\, \Sigma_{k\ \mathrm{connected\ to\ club}}\, \mathrm{InterclubU}(\mathrm{club}, k) \end{aligned} \tag{1}$$

$$\begin{aligned} U(x, y) = a_2\, &|\mathrm{files}(y)|\, (1 - |\mathrm{sim}(\mathrm{files}(x), \mathrm{files}(y)) - \mathrm{target}|) \\ &+ a_3\, \mathrm{sim}(\mathrm{unsucessfulQueries}(x), \mathrm{files}(y)) \\ &+ a_4\, \mathrm{distance}(x, y) \\ &+ a_5\, \mathrm{reputation}(y) \end{aligned} \tag{2}$$

The formulas (1) and (2) state that the evaluating Leaf *x* calculates the utility it gains from all nodes *y*, which are members of the club, and the utility it gains through the interclub connections. The variables a_1 to a_5 are weights that determine the relative importance of the different elements of the utility function. The function *files(z)* represents the content of a node *z*. The similarity function *sim()* is an information retrieval method as described in (Asvanund et al. [5]) and results in a value between zero (not similar) and one (similar). It operates on two sets of files by comparing every element of the first set to every element of the second set and returns the average similarity value. The variable *target* is defined (as described above) as the target similarity between the file collections that a node wants to achieve. The function *InterclubU()* will be described in the following paragraph.

5.4 Utility Function for Interclub Connections

Our topology formation algorithm considers utility function (3) to determine the utility that a club gains by making a connection to another club. Since the establishment of a connection between clubs follows the Gnutella 0.6 protocol specifications, the connection can only occur between Ultrapeers. Therefore, the Ultrapeer's properties are the determining factors in the decision to open a connection. The so-called Interclub utility is defined as:

$$\begin{aligned} \mathrm{InterclubU}(c, k) = (&b_1\, \mathrm{numberOfLeafs}(k) \\ &+ b_2\, \mathrm{files}(k)\ ) \\ &+ b_3\, \mathrm{distance}(c, k) \\ &+ b_4\, \mathrm{reputation}(k) \end{aligned} \tag{3}$$

The variables b_1 to b_4 of the utility function (3) are weights, similar to a_1 to a_5. The utility function has four elements. First, the number of connected Leafs, which is an estimate for the club k's likelihood of fulfilling club c's informational needs. Second, the amount of content *files(k)* offered by club k, which is also an indicator for the likelihood of fulfilling club c's informational needs. Third, the distance of the two clubs' Ultrapeers *distance()*. Since the

distance function is a part of the Leaf utility function, Leafs are likely to be close to the Ultrapeer in the underlying physical network structure as well. Therefore, the distance between the Ultrapeer will give a good estimate of the average distance between the two club's nodes. The last and most important element is the club's reputation. Since, as described above, the Leaf utility function leads to an aggregation of trustworthy nodes in trustworthy clubs and untrustworthy nodes in untrustworthy clubs, this element of the interclub utility function leads the evaluating club to connect to a club with the same amount of trust as itself.

Summarizing the purpose of the inter-club utility function, it can be stated that the connected Ultrapeers will most likely be able to answer the club's queries, be close to the club's nodes, leading to fast response and download times and be trustworthy. This should lead to a very positive overall effect on the whole network, at least for cooperative nodes.

5.5 Disregarding Bandwidth

Estimating a node's bandwidth is inherently difficult and costly. For testing the actual bandwidth of a node, the link needs to be filled with traffic of some form, even if it is only for a short time. Therefore, most P2P applications allow the user to enter his connection type and use this information to estimate the available bandwidth. This, however, is very inaccurate and can be easily misused. Selfish users given the choice of setting the connection type will enter a connection type that gives them the highest benefit, e.g. DSL users who pay for byte volume might enter modem as the connection type to limit the cost for traffic. Because of these issues, our topology formation algorithm disregards bandwidth as a factor. It gives selfish users fewer possibilities to gain unfair advantages.

6 Simulation of a GNet

In order to investigate the performance of our algorithm compared to the original club algorithm, we programmed a simulation of a Gnutella network. For this, we took the simulator used in (Asvanund et al. [5] and Bagla & Kapadia [6]) as a reference implementation. It was generously provided by Ramayya Krishnan. The new version of the simulator, enhanced to be able to run our algorithm, is written in Java and based on the simulation framework J-Sim (Tyan [21]). Additionally, the open-source packet "Java 2D Graph" has been used for some minor calculations [7]. For logging purposes, log4j of the Apache Software Foundation is used [15].
The simulator has two parts, one for the creation of a scenario, the other for running the simulation of the previously created scenario. The split in two applications has the advantage that created scenarios can be reused and exchanged between users. It also allows a manual analysis and manipulation of the created scenarios. The scenario application creates a set of input files. The simulation application reads them in and then starts a simulation run, executing the events as specified in the scenario files. Each simulation run is split into several rounds, as specified in the input files. A round consists of two phases, the evolution phase, in which Leafs and Ultrapeers make new connections and discard old ones, and the query phase, in which the nodes issue Queries for files they are interested in and return QueryHits. The simulator does not model file transfers, only the topology issues and querying. The reason is our assumption that good performance in the resource discovery indicates good performance in the file transfers as well. If many well-trusted locations of a file can be found and they respond with a low latency, then the download of that particular file should be reliable and fast. Therefore, after the query phase, it is assumed that all files for which locations have been found are being downloaded completely and added to the node's file collection. Whether those files will be shared in the next round depends on the node's settings. The simulation results in two datasets with statistics, one for each node per round and one for the whole simulation.

6.1 Creation of Simulation Scenarios

A simulation scenario description consists of text files describing nodes, files, events, and some global parameters. They are generated according to the values given as arguments to the scenario creation application.

The `nodes.txt` file contains information about each node. The most important is the node ID, a global identifier, the files initially shared by the node, information whether the node starts as an Ultrapeer, can become an Ultrapeer during the simulation, and whether it is a freerider or shares files. As described in Bagla & Kapadia [6], a statistical analysis of Gnutella network traffic has shown that 42% of nodes never share files. Therefore, the 42% of all nodes are not assigned files and labeled freeriders in our simulation. The remaining nodes are assigned a certain number of files depending on a long right tail distribution with an average of 270 files. This is modeled by a Weibull distribution with a shape factor of 0.576. The nature of this function leads to the situation that a node can start with zero files without being a freerider. In this case, the node will share any files that it acquires after a query phase. The file `nodes.txt` also contains the TTL that the node's Queries have and a set of coordinates. These coordinates determine the node's position on the network grid, which simulates the underlying physical network structure. The number of club and interclub connections a node can keep open is also specified here. Another important factor is the initial reputation, which is a random number equal to or smaller than the number of files shared. In addition to this, the file `nodes.txt` contains a list of file ID numbers. In the simulator, the similarity between two files is defined in terms of the distance of their ID numbers. Two files with IDs 15 and 16 are very similar, 2 files with IDs 15 and 1600 are not similar. Also, a number of file seeds is chosen, which is normally distributed with a mean of three. The file seeds represent the interest of users for a file type (e.g. type of music). Then, the file IDs are drawn according to a normal distribution around the file seed IDs, until the targeted number of shared files has been reached.

The `files.txt` file describes the content files existing in the modeled Gnutella network. Each content file has a file ID and a size. The maximum number of existing files is defined by a constant, which is a factor that is multiplied by the number of existing nodes. The factor is set to 80 files per node. If a higher number is chosen, the average similarity between two sets of files will be lower, because the IDs of files that two nodes own are randomly chosen from a greater range and the similarity depends on the distance of the file IDs.

The `events.txt` file describes the events that will be sent to nodes in the network. There are two types of events, Evolutions and Queries. An Evolution event causes a node to renegotiate its connections. A Query event specifies the time within the simulation when it is issued and the file ID that the issuing node is looking for. Which file IDs are queried depends on the files a node has. All queried file IDs are normally distributed around IDs the node already owns. If the node shares no files, a random existing file ID is chosen as a file interest and the queried IDs are normally distributed around this file ID. The number of queried files follows a long right tail distribution with a mean of 13, as found in the analysis in Bagla & Kapadia. This is also modeled by a Weibull distribution with a shape factor of 0.576.

The `topology.txt` file contains the initial topology when the simulation starts. The file is empty in this version of the simulator, because the simulation starts with an evolution phase. It is nevertheless possible to manually specify a topology, which would be honored by the simulator.

The `globals.txt` file is used for global system variables, such as finish time and network grid size. Network grid size describes the size of the simulated physical network. Another variable defines how many Ultrapeers are allowed in the system. For our simulations, the value is set to 110%, allowing 10% more Ultrapeers than necessary. This way, all Leafs have the chance to use the maximum number of Ultrapeer connections. This leeway also allows for some competition between Ultrapeers. Ultrapeers who are attractive to nodes will be able to fill their open Leaf connection slots, whereas Ultrapeers who are not attractive will not find Leafs to fill their slots. The attractiveness of an Ultrapeer is determined by its Leaf and interclub connections.

6.2 Running the Simulator

The simulator allows specifying the utility model and whether or not to allow *Ultrapeer Status Transitions*. The utility model can be either `CMU` (Bagla & Kapadia [6], Asvanund et al.[5]) or `IU`. The switch *Ultrapeer Status Transitions* determines whether eligible Leafs can become Ultrapeers and vice versa, if the network needs more Leafs (i.e. there are Leafs who cannot find a club to join) or Ultrapeers (i.e. there are Ultrapeers who cannot find Leafs to join their club). However, a node will only change from or into an Ultrapeer if it could not establish these connections for two rounds.

6.3 The Evolution Phase of the Simulation

In the evolution phase, the nodes renegotiate the network topology. Every connection negotiation uses a three-way handshake as shown in Fig. 2. Each node will attempt to connect to a certain number of Ultrapeers per round, three if the node is a Leaf and five if it is an Ultrapeer. These numbers have been chosen according to the GNet specification.

An active node picks a random Ultrapeer to start connection negotiations. There are two different ways how the connection negotiations are performed. It depends on the type of node. If the node is an Ultrapeer, it will first send a *Connection Request* to another Ultrapeer. If the contacted Ultrapeer has a free connection slot for another Ultrapeer, it will send an *Invitation Message* to the Ultrapeer in question, otherwise it will only send an *Invitation Message* if the replacement of the weakest connected Ultrapeer results in an increased sum of interclub utilities. The active node receiving the *Invitation Message*, decides in the same way to send or not to send a *Confirmation Message*. When a *Confirmation Message* is sent, the sending Ultrapeer will add the receiving Ultrapeer to its list of connected Ultrapeers and the receiving Ultrapeer will add the sending Ultrapeer to its list of connected Ultrapeers. If there are no free Ultrapeer slots, a *Connection Close Message* will be sent to the weakest connected Ultrapeer. In general, if the situation has changed, a *Connection Close Message* can be sent anytime during the protocol.

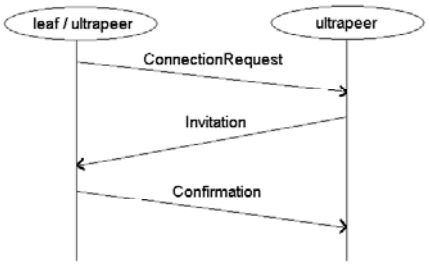

Fig. 2 The connection negotiation uses a 3-way handshake

If the active node is a Leaf, it will send a *Connection Request* to the Ultrapeer. If the contacted Ultrapeer has a free connection slot for a Leaf, it will return an *Invitation Message*. Otherwise, it will only send an *Invitation Message* if replacing the weakest connected Leaf results in a higher club utility. This means that all possible club configurations including the new Leaf and excluding one of the connected Leafs have to be evaluated (i.e. calculating the club utility). The Leaf receiving the invitation will send a *Confirmation Message* if the utility it would gain from this club is higher than the lowest utility of connected clubs or if it has a free connection slot. Otherwise, it will send a *Connection Close Message* to the Ultrapeer. If it accepts, it adds the Ultrapeer to the list of connected Ultrapeers, possibly closing the connection to a weaker club. Once an Ultrapeer has been added, it sends a list of its shared files to the

Ultrapeer, so that the information can be used in subsequent connection negotiations. This list is also needed to simulate the Query Routing Protocol of the Gnutella 0.6 protocol.

6.4 The Query Phase of the Simulation

After the topology for the round has been negotiated during the evolution phase, the nodes will send Querys as specified in the Simulation Scenario. The querying process is handled as defined by the Gnutella protocol. Additionally, a reputation system is modeled in a simplified way. Nodes earn one reputation point by forwarding a query hit and lose five reputation points by generating them. It is assumed that a mechanism exists to prevent cheating. Therefore, all simulated nodes act completely honest. The information about issued and forwarded queries and query hits is stored and used to calculate the metrics at the end of the simulation.

7 Analysis of the Simulation Results

To prove that our algorithm, IUTopForm, provides more incentives for users to cooperate than the original club algorithm, we compared them using the same scenario, with the same settings. We created a scenario with 1000 nodes, 89 of them being Ultrapeers, the others Leafs. In a simulated period of 100,000 time units, 5 rounds are completed, each starting with an evolution phase and ending with a query phase. Each node attempts to become member of three clubs, each club has a maximum of 31 Leafs and one Ultrapeer as members. Each club is connected to up to five other clubs. Query messages are sent with a TTL of four. The targeted similarity level between two file collections is set to 80 percent.

In order to show the effectiveness (close proximity to the requested content in the overlay topology and a large distance to untrustworthy nodes) of our incentive scheme (expressed through the utility functions), we consider the distribution of reputation values of nodes. Since nodes earn reputation by providing services to other nodes (i.e. cooperating), the average reputation value is used to distinguish between cooperative and non-cooperative nodes. From a node's average reputation value, it can be determined whether a node is cooperative or not.

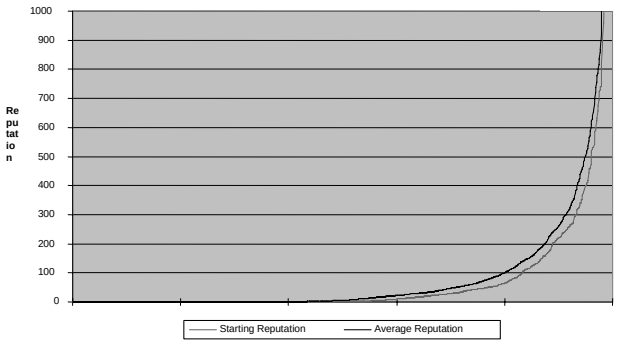

Fig. 3 Reputation distribution

Fig. 3 shows the starting and average (final) reputation value of nodes that belong to five different groups. All nodes are sorted out according to their reputation value and, then, split into five groups. Each group represents a quintile of the entire set of nodes. Initially, all nodes have been given a starting reputation value, according to a long tail distribution. Consequently, the first three quintiles have very low reputation values. Only the last two quintiles can be seen as

partially and, respectively, fully cooperative. The average (final) reputation value, which represents the reputation value of a node within a quintile at the end of the simulation, changed only slightly compared to the initial value.

Close Proximity of the Requested Content. To demonstrate that requested content is closer for cooperative nodes than for uncooperative nodes, we measure how many of the received query hits will be answered from nodes of the club of which the requesting node is a member.

As Fig. 4 illustrates, our algorithm achieves that cooperating nodes get better performance than using the CMU algorithm. Fully cooperating nodes clearly get an improvement in response rates. They participate in clubs, which can satisfy their file needs. All other nodes face lower performance if our approach is applied. Those nodes are penalized for not cooperating sufficiently. Since cooperation is the only way to improve bad performance, our approach forces peers to increase cooperation or live with even worse response rates than in the CMU approach.

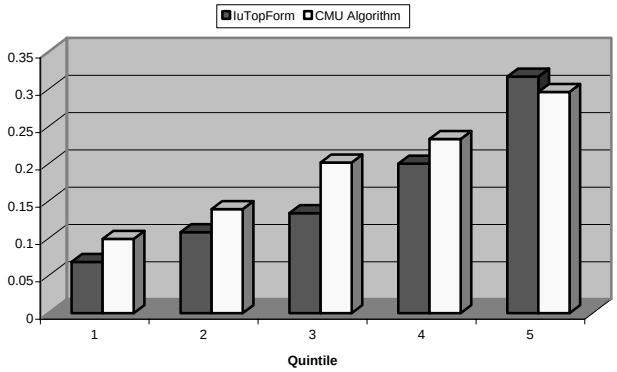

Fig. 4 Number of positive responses from directedly connected clubs

If one considers not only the connected clubs but also all clubs that a node can reach, a similar picture develops (Fig. 5). The weaker four quintiles receive worse service, receiving fewer responses to their requests. Cooperating nodes perform better under the IUTopForm approach, i.e. the strongest quintile of nodes benefits from our incentive scheme.

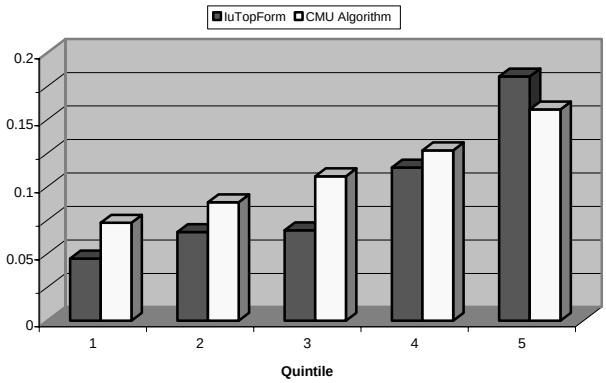

Fig. 5 Number of positive responses from all clubs

Large Distance to Untrustworthy Nodes. The metric that is used to describe the distance of a node to another node is the reputation difference. The reputation difference is calculated by subtracting the average reputation value of all connected nodes from the node's own reputation value and, then, taking the absolute value of the result. Table 1 shows the simulation results for this metric for all three types of connections between nodes.

Table 1 Our algorithm homogenizes the topology's reputation distribution

Average Reputation Difference between …	IUTopForm Approach	CMU Approach
A Leaf and its Ultrapeers	135.685	150.671
A Ultrapeer and its Leafs	106.735	112.430
A Ultrapeer and its connected Ultrapeers	77.081	134.471

As listed in Table 1, this metric shows a decrease in reputation difference for all possible connection types: A clear 11% decrease between a Leaf and its Ultrapeers, an indecisive shift of 5% within clubs, and a major decrease of 74% between Ultrapeers.

The situation within clubs remains the same, because the original algorithm already took the number of shared files into account. Since this number is roughly proportional to the reputation in this simulation. However, the introduction of the reputation term into the utility function did yield significant improvements for the other interclub connections.

8 Conclusion

Within this paper, we introduced a new incentive scheme that is used for a new, market-managed topology formation algorithm. Our algorithm, which has been inspired by the approach of Asvanund et al., adds the additional dimension of reputation to the incentive scheme. In addition to this, our topology formation algorithm incorporates a new model for predicting future information demands, argues not to include the consideration of bandwidth, and proposes two new utility functions (i.e. a utility function for Leaf nodes and a utility function for interclub connections).

The simulation results show that nodes with similar reputation value are close to each other in the topology of the P2P network (those nodes are clustered within the same club). Our market-managed topology formation algorithm has had the effect that nodes with a high reputation value receive better service than under the algorithm of Asvanund et al.. This shows that our algorithm forces users either to cooperate or to tolerate even reduced file-sharing performance.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] Abdul-Rahman, A. and Hailes, S., "Supporting Trust in Virtual Communities", In *IEEE Proceedings of theHawaii International Conference on System Sciences,*Maui, Hawaii, January 4-7, 2000.

[3] Aberer, K. & Despotovic, Z., "Maximum Likelihood Estimation of Peers' Performance in P2P Networks," EPFL - Swiss Federal Institute of Technology, 2004.

[4] Abrams, Z., McGrew, R. & Plotkin, S., "Keeping Peers Honest in EigenTrust," Stanford University, 2004.

[5] Asvanund, A., Bagla, S., Kapadia, M., Krishnan, R., Smith, M., & Telang, R., "Intelligent Club Management in Peer-to-Peer Networks," Carnegie Mellon, Heinz School of Public Policy and Management & Information Networking Institute, 2003.

[6] Bagla, S. & Kapadia, M. H., "Peer-To-Peer Self-Organizing Communities," Carnegie Mellon University, Information Networking Institute, 2003.

[7] Brookshaw, L., "Java 2D Graph," Retrieved on September 28th, 2004, from http://www.sci.usq.edu.au/staff/leighb/graph/source/SpecialFunction.java, 1996.

[8] Cohen, B., "Incentives Build Robustness in Bittorrent," In 1st Workshop on Economics of Peer-to-Peer Systems, June 2003.

[9] Dutta, D., Goel, A., Govindan, R. & Zhang, H., "The Design of A Distributed Rating Scheme for Peer-to-Peer Systems," University of Southern California / Stanford University, 2003.

[10] Feldman M., C. Papadimitriou, J. Chuang, & I. Stoica, "Free-Riding and Whitewashing in Peer-to-Peer Systems," ACM SIGCOMM04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS), August 2004.

[11] Hee Lee, C. & Hwang, J, "Agent-based Modeling for Differentiated Admission in P2P Systems Using Evolutionary Game Theory Focused on Ownership Reputation," Seoul National University / Syracuse University, 2004.

[12] Jennings, T., "Fido and FidoNet," Retrieved on August 19th, 2004, from http://www.wps.com/FidoNet/ .

[13] Kirk, P., "Gnutella 0.6 - Defining a Standard," Retrieved on August 19th, 2004, from http://rfc-gnutella.sourceforge.net/developer/index.html, 2003.

[14] Kung, H.T. & Wu, C., "Differentiated Admission for Peer-to-Peer Systems: Incentivizing Peers to Contribute their Resources," Harvard University/ Academia Sinica, 2003.

[15] log4j project, "Logging Services - log4j," Apache Software Foundation. Retrieved on September 28th, 2004, from http://logging.apache.org/log4j/docs/ , 2003.

[16] McManus, S., A short history of file sharing. Retrieved on August 19th, 2004, from http://www.sean.co.uk/a/musicjournalism/var/historyoffilesharing.shtm , 2003.

[17] Mello, J., "File Sharers Deserting Kazaa's FastTrack Protocol," Retrieved on August 18th, 2004, from http://www.technewsworld.com/story/34305.html, 2004.

[18] RFC-Gnutella 0.6, "RFC-Gnutella 0.6," Gnutella Developers Forum. Retrieved on August 19th, 2004, from http://rfc-gnutella.sourceforge.net/developer/testing/index.html , 2004.

[19] Ritter, J., "Why Gnutella Can't Scale," No, Really.. Retrieved on Aufust, 26th, 2004, from http://www.darkridge.com/~jpr5/doc/gnutella.html , 2001.

[20] Rohrs, "Query Routing for the Gnutella Network," Retrieved on August 26th, 2004, from http://rfc-gnutella.sourceforge.net/src/qrp.html , 2001.

[21] Tyan, H., "Design, Realization and Evaluation of a Component-Based Compositional Software Architecture for Network Simulation," Ohio State University, 2002.

[22] Usenet History, "Usenet Software: History and Sources," interbulletin.com. Retrieved on August 19th, 2004, from http://news.interbulletin.com/usenet_his.html.

[23] Walsh, K., & Sirer, E.G., "Fighting Peer-to-Peer SPAM and Decoys with Object Reputation," Proceedings of the Third Workshop on the Economics of Peer-to-Peer Systems (p2pecon), Philadelphia, USA, 2005.

[24] Wikipedia, "Gnutella," Wikipedia. Retrieved on August 19th, 2004, from http://en.wikipedia.org/wiki/Gnutella , 2004.

[25] Wikipedia, "Peer-to-peer," Wikipedia. Retrieved on August 18th, 2004, from http://en.wikipedia.org/wiki/Peer-to-peer, 2004.

[26] Yang, M., Chen, H., Zhao B.Y., Dai, Y., & Zhang, Z., "Deployment of a Large-Scale Peer-to-Peer Social Network," WORLDS04, 2004.