Peer-to-Peer Netw Appl (2009) 2:37–59
DOI 10.1007/s12083-008-0018-2

Adaptive neighborhood selection in peer-to-peer networks based on content similarity and reputation

Ioannis Pogkas · Vassil Kriakov · Zhongqiang Chen · Alex Delis

Received: 14 April 2008 / Accepted: 15 October 2008 / Published online: 15 January 2009

Abstract To address the two most critical issues in P2P file-sharing systems: efficient information discovery and authentic data acquisition, we propose a *Gnutella*-like file-sharing protocol termed *Adaptive Gnutella Protocol* (*AGP*) that not only improves the querying efficiency in a P2P network but also enhances the quality of search results at the same time. The reputation scheme in the proposed *AGP* evaluates the credibility of peers based on their contributions to P2P services and subsequently clusters nodes together according to their reputation and shared content, essentially transforming the P2P overlay network into a topology with collaborative and reputed nodes as its core. By detecting malicious peers as well as free-riders and eventually pushing them to the edge of the overlay network, our *AGP* propagates search queries mainly within the core of the topology, accelerating the information discovery process. Furthermore, the clustering of nodes based on authentic and similar content in our *AGP* also improves the quality of search results. We have implemented the *AGP* with the *PeerSim* simulation engine and conducted thorough experiments on diverse network topologies and various mixtures of honest/dishonest nodes to demonstrate improvements in topology transformation, query efficiency, and search quality by our *AGP*.

Keywords Peer-to-peer · Adaptive topology · Reputation

I. Pogkas · A. Delis
University of Athens, Athens 15784, Greece

I. Pogkas
e-mail: i.pogkas@di.uoa.gr

A. Delis
e-mail: ad@di.uoa.gr

V. Kriakov (✉)
Polytechnic Institute of New York University,
Brooklyn, NY 11201, USA
e-mail: vassil@cis.poly.edu

Z. Chen
Yahoo! Inc., Santa Clara, CA 95054, USA
e-mail: zqchen@yahoo-inc.com

1 Introduction

Peer-to-peer (P2P) networks are extensively used today for sharing information and improving information dissemination. There are two significant problems that exist in P2P networks: (1) discovering information efficiently and (2) obtaining authentic information. The problems have been studied extensively but mostly in isolation. In this paper, we introduce a novel protocol named *Adaptive Gnutella Protocol* (*AGP*) that addresses both problems simultaneously, resulting in improved P2P service.

1.1 Topology & query efficiency

Peer-to-peer networks consist of nodes and connections between the nodes. As fully connected networks are inefficient (the number of connections grows quadratically with the number of nodes), overlay networks are often used. Nodes in overlay networks are connected through virtual links that correspond to multiple physical links in the underlying infrastructure. In that respect, two types of overlay networks exist, structured and unstructured. *Structured* P2P networks traditionally employ a globally-consistent indexing

scheme to ensure that, in a network of n nodes, a query can be routed within $O(log(n))$ hops to a peer that offers the requested content [28]. Unfortunately, the capability of such networks to efficiently support complex queries and to form communities is limited [17]. Moreover, the cost of maintaining the overlay structure and routing tables with document indexes is also increased due to continuously departing and joining nodes [6]. Therefore, structured P2P networks, although able to provide query performance guarantees, fail to provide a robust solution for the problem of discovering information efficiently.

On the other hand, in *unstructured* networks, like *Gnutella*, global consistency is forgone in favour of flexibility. The overlay topology is organized into a random graph, improving the decentralized maintenance of the network and eliminating the severe problems caused by a high node churn rate. Search capabilities in unstructured networks are supported through mechanisms such as breadth-first-search, depth-first-search, and Random Walks [30], which are oblivious to the overlay structure. As unstructured P2P networks do not use indexes based on content descriptions for the query propagation, they require low memory overhead for query processing and can easily handle more complex queries. This is done at the expense of inefficient lookups when searching for unpopular content [6]. Often, the solution is to form well-connected communities with similar content interests. In this paper, we propose a protocol for improving query performance in unstructured P2P networks by carefully organizing the topology so that long-lasting connections are established between nodes with similar information interests.

1.2 Reputation & information authenticity

Although the problem of discovering information may have an obvious solution, such as augmenting the topology (we will see that there are many ways to do so), the second problem of obtaining authentic information is more difficult to solve. The situation is exacerbated by the fact that P2P networks are open communities, allowing anyone to join and to subsequently serve any content. The complete decentralization in P2P networks means that there is no central authority for content and identity authentication. Current file-sharing approaches for both structured and unstructured networks suffer from *dishonest* nodes that propagate low-quality or dangerous content (e.g. malicious software or malware). In such a situation, the final verdict on content quality is ultimately delivered by the user at the receiving end. Malicious content, such as viruses, worms, trojan horses and key loggers, is more difficult to discern, requiring end users to rely on security protection software.

However, there is another type of node that negatively affects a P2P network—*free-riders* [14] that make use of the network resources without contributing any content or services in exchange. As the idea of P2P networks is to *share* information, nodes that do not share information or services are parasitic to the network. Therefore, it would be of value if such nodes could be identified and appropriately dealt with. There are two additional aspects of difficulty in maintaining a P2P network of honest peers with authentic content. Firstly, multiple nodes may collude in an attempt to present themselves as authentic information providers. Generally, if such nodes outnumber the honest peers that are interested in the same information, it is not possible to discern their malicious intent. The second aspect is that malicious nodes may alternate between honest and dishonest behaviour. Specifically, a node may behave honestly but could conduct dishonest activities after it gains sufficient trust from other peers. In this paper we propose a reputation based scheme which attempts to discern such malicious peers by monitoring their historic behaviour, as well as their current behaviour.

The necessary properties of a functioning reputation system are discussed in detail in [21]. Fundamentally, a reputation system assists agents in choosing a reliable peer to transact with. The choice comes from a pool of possible providers. A reputation system collects information about a peer's behaviour, scores and ranks the peer using specific metrics, takes action against the malicious peers and rewards the honest ones. The ultimate goal of a reputation system is not only to find and punish the malicious peers, but also to reduce the maintenance cost for the honest ones.

The design of a reputation system depends on the defined assumptions on the user/peer behaviour and the network environment. In our proposal, we assume that the users require no guarantee about the reliability of the system services. On the other hand, nodes require privacy protection and anonymity. This is achieved by using globally unique identifiers to represent nodes in the network. In addition, when two nodes transact, the source is authenticated using public/private key encryption. Although this leaves the system vulnerable to a man-in-the-middle attack, our scheme can easily be expanded to use certificates signed by a pre-trusted authority. We do not explore this extension here but do assume the existence of pre-trusted GWebCaches in order to boot-strap joining nodes. Furthermore, we assume that peers can easily enter and leave the network, usually without graceful disconnect, resulting in high node *churn rate*.

In our reputation system four types of dishonest peers define the threat model: (a) free-riders that use system services without forwarding queries or contributing resources, (b) malicious peers that attempt to cause harm either to a number of network members or to the network as a whole, (c) peers that provide low-quality content, and (d) peers whose behaviour dynamically alternates between models (a) and (b) above. Usually the malicious nodes distribute corrupted audio files on music-sharing networks [1] or disseminate virus infected files [2]. A special category of peers are moles. Moles provide misinformation, in order to promote specific malicious peers. Mole peers are confronted with weighted evaluations from a number of trusted peers, and by giving more weight to the peer's own experience. Our protocol considers the dynamic peer behaviour, not only for the malicious nodes but also for the free-riders, including no-forwarders – a type of free-riders that, in addition to not sharing content, do not provide any query forwarding services. When dishonest peers misbehave, they are punished with low reputation scores and their reputations keep decreasing if their misbehaviour persists, causing the topology adaptation mechanism in our proposed *AGP* to eventually push them to the edge of the network. It is very difficult to completely prevent peer collusion because the nodes take random identities and are placed at random positions in the network. However, *AGP* uses neighbor recommendations, making collusion more difficult to achieve. A whitewashing attack occurs when a peer returns to the P2P system as a newcomer with a newly generated identifier [21]. *AGP* deals with whitewashing attacks by assigning low reputation scores to newly joined nodes so that peers are encouraged to gain reputation by providing services to the system or rejoining the network with identical IDs. In addition, historical behaviour of peers is also taken into account in the reputation evaluation, causing nodes that launch whitewashing attacks to obtain low reputation due to their lack of history. On the other hand, a malicious node also obtains a bad reputation score if it comes back to the system with the same identity due to its bad history.

Denial of service (DoS) attacks occur when malicious peers consume large amounts of physical resources to completely disrupt services. Our protocol takes the following considerations in order to ameliorate such situations: (a) each node has a limited number of network connections and message queue size and (b) when a node becomes overloaded it informs its neighbors. If a neighbor violates the notifications of overloaded nodes, its reputation is severely penalized. In addition, we use a simple mitigation defense strategy: each node has a low and high water levels on its resources (bandwidth, CPU and memory) that can be dedicated to serve other peers. When the high water level is exceeded, the peer stops to provide services until the utilization of resources falls below the low water level. In this manner, peers defend themselves from coordinated attacks launched by colluding malicious nodes.

1.3 Our contributions

To address the above problems, we propose a novel protocol termed *Adaptive Gnutella Protocol* (*AGP*) that is integrated into *Gnutella*-like unstructured P2P networks to effectively cluster *honest* peers together and transform the overlay network into a topology with collaborative and trusted nodes as its core. *AGP* also provides message-forwarding services to more effectively materialize queries. In our proposed protocol, nodes are incentivized to use their network/hardware resources in order to forward queries to their neighbors. *AGP* creates new connections between counterparts that provide similar content and restricts the connections with the malicious nodes.

Our protocol proceeds in a step-wise manner to attain its goals: initially, peers use a reputation scheme to evaluate every neighbor as a service provider. The dishonest neighbors which provide low-grade/malicious services and do not contribute their resources will eventually receive low reputation score. As peers divide their available bandwidth to serve neighbors in a way proportional to their reputation evaluation, dishonest neighbors receive degraded service. In addition, all the peers connected with direct virtual links exchange descriptions about their content. Using the above information our protocol exploits both content similarity and reputation monitoring to rearrange the overlay network connections and achieve enhanced topological formation among peers. Finally, our protocol uses a search mechanism that takes into account content similarity *and* reputation in materializing better results.

Through experimentation with the *AGP* implementation, we establish that, over time, the network topology improves as the peers are able to locate counterparts with similar content and better reputation. Furthermore, we show that even in the presence of a large number of malicious nodes, *honest* peers increase their connectivity with other honest nodes and manage to avoid the *dishonest* ones. Our simulations include nodes which alternate their behaviour between honest and dishonest in attempt to gain and misuse reputation. Our experiments show that the malicious nodes are pushed to the edge of the overlay network as

honest peers drop the connections with them. Finally, compared to the *Gnutella* Dynamic Query Protocol (DQP) [3], our proposed search algorithm returns improved query responses using less network resources. As a result, *AGP* improves the overall P2P network operation and provides better query results while consuming much less network resources.

The remainder of the paper is organized as follows: Section 2 provides the basic features of the proposed protocol elements. Section 3 discusses the design of *AGP* and Section 4 presents our methodology on testing *AGP* and the evaluation results. Section 5 compares *AGP* to related work. Finally, in Section 6 we present the conclusions and future work.

2 *AGP* features & messages

The proposed *AGP* consists of five modules as shown in Fig. 1. The Content Exchange and Reputation Management modules maintain peer information relating to quality and similarity of resources, as well as peer behaviour, benign or malicious. This information is leveraged by the Topology Adaptation module in making decisions when establishing or dropping connections. The above information is also pertinent to query routing as deemed necessary by the Searching Mechanism module. The Cache module maintains and synchronizes the information acquired by the other four modules. Each module will be discussed in further detail in Section 3. The specific characteristics of the peers, the messaging and the resources management are described below.

Peer characteristics Each node provides encryption, decryption and hashing services. This way the peer can check the integrity of the downloaded content, and can protect its messages using cryptographic services.

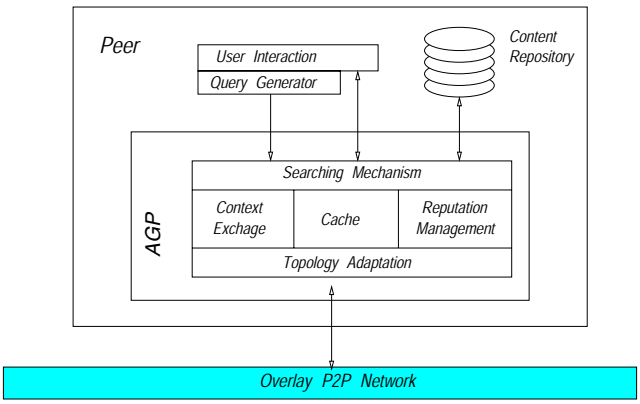


Fig. 1 The architecture of *AGP*-protocol and its modules

To achieve this, our protocol uses the *MD5* hashing algorithm [24] and RSA-cryptography [25]. Provided that a peer maintains a *public* key and a *private* key, messages are encrypted at the source node using the public key of the destination node. The latter is the only one which can decrypt messages successfully, using its private key. Nodes feature a unique *ID* that helps them maintain consistent identity. This *ID* is produced by hashing the node's private key. In this regard, every node is identified with an *ID* that can be generated locally, transferred easily, but cannot be easily stolen or created. It is possible to use "zero-knowledge proof" methods [10, 12] in order for a peer A to ascertain the validity of the ID and public key generated by peer B. Due to the complexity of the zero-knowledge proof techniques and their orthogonality to *AGP*, we leave their analysis for future work.

As depicted in Fig. 1, each node contains a *Query Generator*, discussed in Section 3.4, used to launch new queries. We consider that peers are interested in specific content categories or file keywords and the submitted queries are for similar content. In reality, an application form will help a user designate her query, parse it into specific keywords, form the query message(s) and invoke the searching mechanism by shipping messages to appropriate sets of neighbor nodes. In addition, for certain file types (e.g., mp3 files), keywords may be generated automatically by extracting information from the file. In this paper, we assume that the keywords describing the content are of good quality and they accurately reflect the content of the file. Shareable content, such as music and video files, is stored in the node's *Content Repository* (Fig. 1). For each file, this repository contains an *XML* description, a file-digest created through a hash-function, and the location of the content. A table such as that shown in Table 1 helps organize the above meta-data.

Messages Each message is stamped with the *ID* of the source and destination node. All messages that nodes exchange are tagged at their origin with a globally unique identifier or *GUID* [3], a randomly generated sequence of 16 bytes. This prevents the routing protocol from sending the same message twice to the same node. The transmission of each message is depicted in Figs. 3–6 where Fig. 2 is the legend for the different

Table 1 The data structure for Content Repository

$File_1$	$File_1$ XML description	$File_1$ MD5 digest	$File_1$ path
$File_2$	$File_2$ XML description	$File_2$ MD5 digest	$File_2$ path
$File_3$	$File_3$ XML description	$File_3$ MD5 digest	$File_3$ path
...	...	...	...
$File_N$	$File_N$ XML description	$File_N$ MD5 digest	$File_N$ path

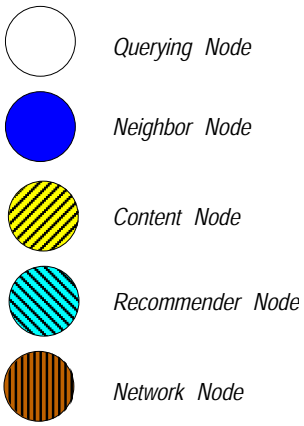


Fig. 2 Node symbols used in various network formations depicted in Figs. 3–6

node types. The messages used by the *AGP* protocol are the following:

- *RequestContent[source = peerID, destination = nodeID]*: informs neighbor *nodeID* that peer *peerID* requests content descriptions (see Fig. 3).
- *SendContent[source=nodeID, destination=peerID, XML-fileDesc*$_1$*, XML-fileDigest*$_1$*, XML-fileDesc*$_2$*, XML-fileDigest*$_2$*, ..., numNeighbors*$_{nodeID}$*, sat]*: is the reply to the *RequestContent* message. This message encapsulates all information regarding the content of node *nodeID*, includes the *XML* description of its files and their corresponding *MD5*-digests used to ascertain that file's integrity. Furthermore, this message informs the querying peer about the number of neighbors *(numNeighbors*$_{nodeID}$*)* that *nodeID* currently has, while *sat* declares whether *nodeID* is satisfied (or not) with its current topology (see Fig. 3).
- *GiveRecomm[sourceID = peerID, destID = node*$_i$*ID]*: The peer *peerID* requests from node *node*$_i$*ID* a set of recommender nodes (see Fig. 4).

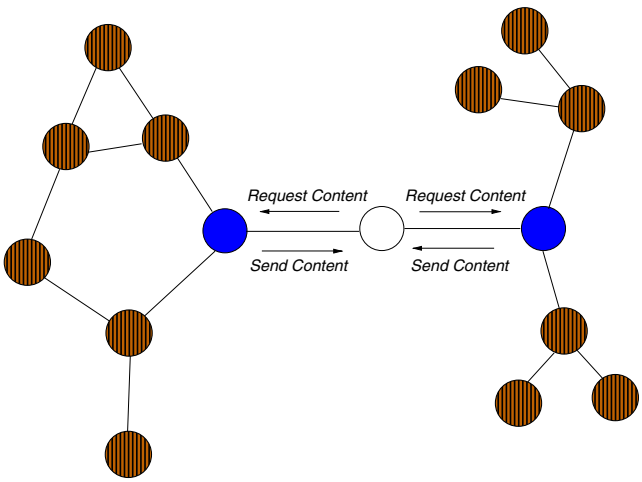


Fig. 3 *AGP*-protocol *RequestContent* and *SendContent* messages are transmitted only between neighbors

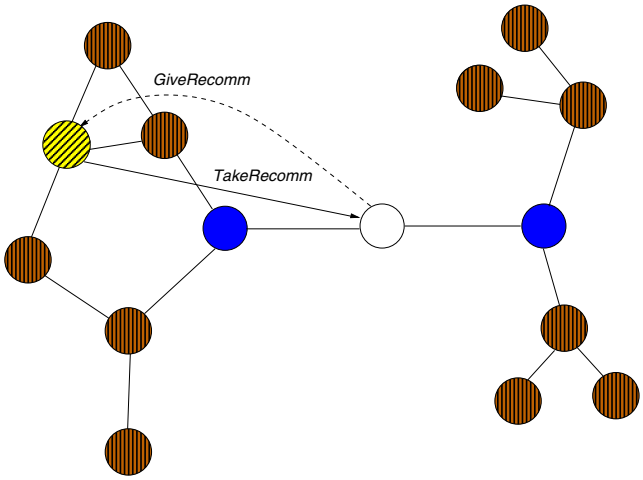


Fig. 4 *AGP*-protocol recommendations are received from recommender nodes which need not be neighbors of the peer in question

- *TakeRecomm[sourceID=node*$_i$*ID, destID=peerID, recommender*$_1$*ID, recommender*$_2$*ID, ...]*: the reply to a *GiveRecomm* message. The node *node*$_i$*ID* provides to peer *peerID* a set of recommender nodes with IDs *recommender*$_1$*ID* , *recommender*$_2$*ID* , etc. (see Fig. 4).
- *Query[sourceID = peerID, string = queryString, RepThreshold]*: a node transmits its query string to its neighbors (see Fig. 5). Forwarding neighbors pass the query on to other peers with similar content and/or reputation higher than *RepThreshold*.
- *Respond[sourceID=responderID, destID=SourcePeerID]*: if a peer contains information which

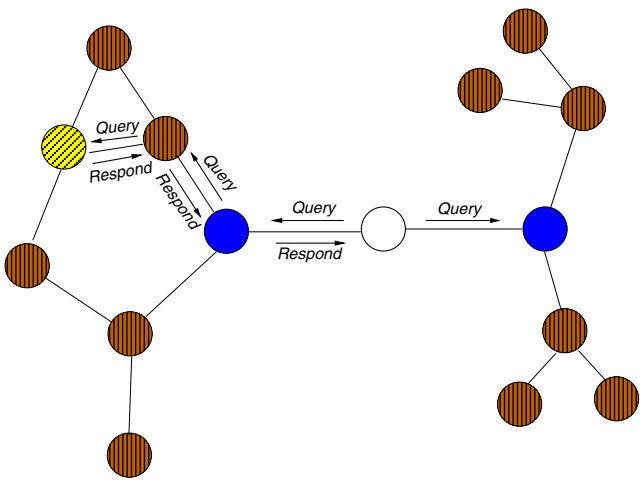


Fig. 5 *AGP*-protocol Query messages are transmitted between neighbors only. Nodes which provide forwarding services will transmit the query to any other peers that might be likely to produce hits

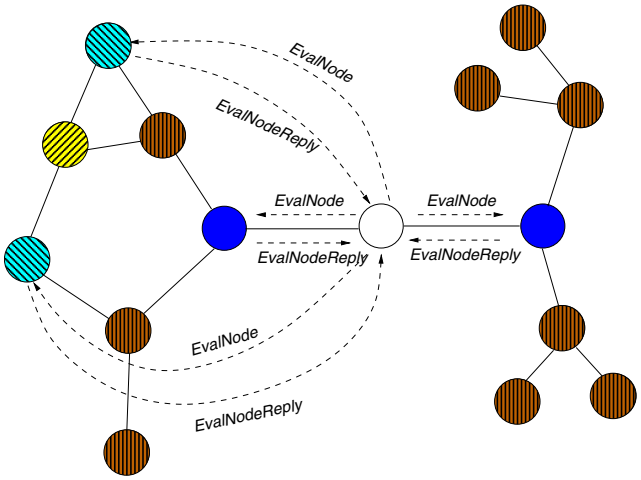


Fig. 6 *AGP*-protocol node evaluations are requested from a subset of the neighbor's recommenders

matches a query, the response is sent back (see Fig. 5). Forwarding peers transmit responses to the node where the query originated.

- *EvalNode[sourceID = peerID, destID = recommender$_k$ID, evalID = node$_i$ID, PK$_{peer}$]*: The peer *peerID* requests from the recommender node *recommender$_k$ID* to provide an evaluation about the node *node$_i$ID* (see Fig. 6). The *peerID* also includes its public key (*PK$_{peer}$*).
- *EvalNodeReply[sourceID = recommender$_k$ID, destID = peerID, evaluation]$_{PK_{peer}}$*: the reply to an *EvalNode* message. The node *recommender$_k$ID* responds to *peerID* with an evaluation (see Fig. 6). The message is encrypted with the public key of *peerID*.
- *Overloaded[sourceID = peerID, destID = neighborNodeID]*: The peer sends this message to all its neighbor nodes, in order to inform them that it is overloaded.
- *Free[sourceID = peerID, destID = neighborNodeID]*: The peer sends this message to all its neighbor nodes, in order to inform them that it is no longer overloaded.

Resource properties Each resource has an *XML* description and a *digest* that is used by the downloader to ascertain the integrity of the resource. For example, an *mp3* file may have the following *XML* description:

```
<mp3>
    <name>Gnutella</name>
    <artist>Artist</artist>
    <album>AGP</album>
    <genre>P2P</genre>
</mp3>
```

The digest is the result of hashing the resource with the *MD5* algorithm.

Content similarity The similarity metric in our protocol is very important and is used by both the topology adaptation module (Section 3.3) and the query routing process (Section 3.4). The reputation scheme identifies dishonest neighbors and assigns them low reputation scores. Eventually, the adaptation process will remove the neighbors that have reputation lower than that of other discovered peers. The candidate peers with the best reputation are sorted based on their content similarity. In this manner, a node will establish connections first with those peers that have the most similar content, creating a cluster of "like-minded" peers. In case that two or more candidates have the same *content similarity*, the node selects one of them randomly. The entire process is described in detail by the topology adaptation algorithm (Section 3.3). The second usage of the similarity metric is in query routing. As we will see in Section 3.4, the nodes always forward queries to peers with similar content, increasing the probability of discovering the information sought by the user.

Content similarity is computed as follows. Suppose that a peer P has a set of XML descriptors XML_P of its files and a neighbor node N has its own set of XML descriptors XML_N. Nodes P and N are connected to each other and have exchanged their content descriptors. In order for peer P to calculate its content similarity with the node N, it parses its XML_P descriptions and creates a set of the ζ (= 100) most popular terms (P_{popSet}). Subsequently, the same process is repeated for the (N_{popSet}) that was offered by node N. As depicted in Algorithm 1, the content similarity score is calculated as the number of identical attributes in the two sets N_{popSet} and P_{popSet}.

Dishonest nodes may advertise false content or content that they do not have in order to allure other peers

Algorithm 1 *similarity*(P_{popSet}, N_{popSet})

```
1:  similarity ← 0
2:  for each file F^P ∈ P_popSet do
3:    for each attribute A of file F^P do
4:      for each file F^N ∈ N_popSet do
5:        for each attribute B of file F^N do
6:          if (A.name == B.name) && (A.value == B.value) then
7:            // Identical attributes with identical values
8:            similarity++
9:          end if
10:       end for
11:     end for
12:   end for
13: end for
14: return similarity
```

to establish a connection. Although this scenario can happen, over time, the reputation of such nodes will decrease either automatically or when the user evaluates the content. It is possible to automatically detect when a rogue peer advertises incorrect information by comparing the hash values for the identical information from other peers. If the hash values are different, or after downloading the content it is established that the actual hash value is different from that advertised, it is likely that the content is fraudulent. Although a Byzantine protocol [5] may be used here to down-grade the reputation of dishonest peers, it requires at least 2/3 of the nodes with the identical content to be honest. This requirement may be too high for less popular content. At present, we leave it to the user to rate the content and consequently, the source of the content. Ultimately, our reputation mechanism (*proRep* metric in Section 3.2.3) will cause the reputation of the dishonest node to rapidly decrease and the connection to the rogue peer will eventually be discarded by the topology adaptation mechanism (Section 3.3).

3 The *AGP* protocol modules

Figure 1 depicts the characteristics of the *AGP*-protocol and shows its functional modules which are:

- the *topology adaptation* module which modifies existing direct overlay links to other peers, creates new connections and drops links with peers that provide low-quality content or poor forwarding services. Essentially, the task of this module is to gracefully adapt the overlay network topology while using input from the rest of the *AGP* modules.
- the *reputation management* module whose objective is to continually monitor the behaviour of discovered nodes. This component evaluates the nodes as content providers, by examining their forwarding services and by taking into account recommendations from third parties. The nodes are assessed based on both their current and past behaviour.
- the *content exchange* module uses the content stored in the node's content repository to advertise it through *SendContent* messages. In a similar fashion, and via *RequestContent* messages, content-descriptions from other peers may be requested. These two types of messages help to update and publish the node's content into the network. More importantly, this module is in charge of updating the data stored in the node's cache with information about the content of other peers.
- the *cache* module maintains all information used by the *AGP*-protocol as well as meta-data provided by other discovered nodes.
- the *searching mechanism* selects the proper set of neighbors to which the query will be forwarded, based on content and reputation criteria. We modified *Gnutella*'s searching algorithm, in order to forward the queries to the neighbors that have similar content and/or higher reputation.

Peers responding to a query are classified into two categories: *a)* those that maintain direct overlay links to the requesting node called *neighbors* and *b)* peers that the node has possibly downloaded data from but do not belong to the *neighbors* group, called *collateral* nodes. We refer to these two categories of peers collectively as the *discovered* nodes. In the remaining of this section, we outline the structure and the operational aspects of the above modules.

3.1 The *content exchange* module

Upon joining the P2P network, a node is initialized by the following steps:

1. Request $maxNumberOfNeighbors$ (see Table 2) random neighbors from a well-known GWebCache [3].
2. Send *RequestContent* message to each neighbor to obtain XML content descriptions.
3. Neighbors respond with a *SendContent* message to the new peer informing it of their content, their satisfaction state (i.e., "satisfied" / "not satisfied") and the number of their neighbors.
4. The initializing peer uses the information from the *SendContent* message to bootstrap its *Topology Adaptation* module and its Cache.

Steps 2–4 above are repeated for all nodes in the *neighbor* group and every time the peer encounters a new collateral node. The information obtained through *SendContent* messages is stored in the cache, discussed in Section 3.5. The *content exchange* module is responsible for keeping the rgi Cache updated and removing obsolete entries.

The content description of nodes is used for two purposes. Firstly, our *topology-adaptation* uses the meta-data stored in the cache to evaluate *content similarity* with other nodes. The peer establishes connections with these nodes in order of *content similarity*. If two or more nodes have the same *content similarity*, the peer selects one of them randomly. Secondly, the

Table 2 *AGP* Parameters: their ranges, default values and properties

Parameter	Default value	Min value	Max value	Static/dynamic
Capacity	196 kb/s	56 kb/s	420 kb/s	Static
minAllocation	28 kb/s	28 kb/s	28 kb/s	Static
SatisfactionThreshold	0.7	0	1	Static or dynamic
maxNumberOfNeighbors	7	2	15	Static
MaxResponses	1	1	User choice	Dynamic
RepThreshold	0.6	0	1	Dynamic
δ ("good" rating reputation adj.)	0.2	0.2	0.2	Static
λ ("bad" rating magnification)	2.5	2.5	2.5	Static
θ ("dangerous" rating magnification)	5	5	5	Static
α_r (recommendation weight)	If stranger, half of *Rep* of the Least trusted known peer, Else *Rep* of the known node	0	1	Dynamic
ν (*nbRep* memory depth)	0.5	0	1	Static
μ (*othersRep* memory depth)	0.5	0	1	Static

proposed *searching algorithm* uses the content descriptions to make better choices when routing queries. As a result, the queries are only transmitted to neighbors that are highly likely to satisfy them.

3.2 The *reputation management* module

The scope of this module is to provide incentives for peers to avoid selfish and malicious behaviour, discriminate the honest from dishonest nodes and punish the latter with low reputation evaluations. A node's reputation is set proportionally to the services that it provides to its peers. When a node forwards an extensive number of query responses, while acting as intermediary and providing high quality content, its reputation score increases.

Nodes use three fundamental metrics to evaluate a peer's reputation:

- *neighbor reputation evaluation (nbRep)*: it evaluates neighbors for their forwarding services. Good evaluation is given when a node forwards many responses back to the peer for its submitted query.
- *provider node reputation evaluation (proRep)*: it evaluates a node as a content-provider. When a node provides high-quality content to a requesting peer, its *proRep* score increases. Conversely, nodes that disseminate unwanted, poor quality or even malicious content are penalized by receiving low *proRep* scores.
- *node reputation evaluation based on recommendations from others (othersRep)*: it evaluates a node using recommendations from collateral nodes or neighbors. This way the peer can capitalize on the experiences of its counterparts in the P2P network.

Our reputation scheme evaluates peers based on their current and past behaviour. For simplicity and efficiency of our model, we avoid a real time metric. We measure the time in epochs that have specific time duration. The current epoch is defined as t and the previous one as $t-1$. The reputation evaluation of a node i that is either a direct neighbor or a collateral counterpart, at epoch t, is defined as $0 \leq Rep_i(t) \leq 1$. We set the three aforementioned metrics so that $0 \leq nbRep_i(t), proRep_i(t), othersRep_i(t) \leq 1$. We combine them into a single value using three factors $0 \leq c_1, c_2, c_3 \leq 1$ for which $c_1+c_2+c_3 = 1$ is always true. As the nature and quality of services continuously change, the value of $Rep_i(t)$ reflects such changes and is computed as:

$$Rep_i(t) = c_1 \cdot nbRep_i(t) + c_2 \cdot proRep_i(t) + c_3 \cdot othersRep_i(t) \quad (1)$$

In the remainder of this section, we give a detailed description of the above metrics calculation. Furthermore, we present the mechanism that incentivizes the nodes to cooperate and share resources.

3.2.1 Incentives for providing query forwarding service

During an epoch, each node uses a portion of its bandwidth capacity[1] (C) to submit its queries. The rest of the available capacity C_{left} is proportionally shared to serve its neighbors. Hence, the node's capacity (C_i) that is devoted to *neighbor i* is:

$$C_i = C_{left} \cdot \frac{nbRep_i}{\sum_{k=1}^{numberOfNeighbors} nbRep_k} \quad (2)$$

where $0 \leq C_i \leq C_{left}$ and $1 \leq i \leq numberOfNeighbors$.

[1] Another metric could be the node's buffer space for outgoing messages, during an epoch.

The neighbors which provide good forwarding services shall receive a higher reputation score. They are incentivized to do so, as the peer allocates a bigger slice of its bandwidth to serve each contributing node (based on Eq. 2).

3.2.2 Computing the nbRep parameter

Upon query submission, a peer uses its cache content and reputation information to forward the query to a subset of the neighbors with the most similar content (Algorithm 1) and highest reputation. When the query responses arrive, every neighbor is evaluated based on the number of provided responses. We note that at this stage the peer does not mind if the responses were sent by neighbors or by third parties (i.e., the neighbors that just forwarded replies back to the requesting peer). The evaluation for each neighbor i with ID ($EvalNeighborID_i$) that returned R_i responses from the total responses ($totalR$) that the node received is:

$$EvalNeighborID_i = \frac{R_i}{totalR} \tag{3}$$

The $nbRep_i$(t) evaluation of each neighbor i also takes into consideration the previous evaluation (during the previous epoch t-1) of the peer. In this respect, neighbors have to provide uninterrupted good forwarding service to maintain a good evaluation score. The current $nbRep_i$ ($0 \leq nbRep_i(t) \leq 1$) is calculated as a front-weighted moving average of the previous values. This gives higher priority to more recent scores, while still taking into account historical behaviour. The $nbRep_i$ value can be maintained iteratively using the following equation:

$$nbRep_i(t) = \nu \cdot nbRep_i(t-1) + (1-\nu) \cdot EvalNeighborID_i \tag{4}$$

where $\nu \in [0, 1]$ controls the "memory depth" with values closer to 0 giving more weight to more recent measurements. By default, $nbRep_i(0) = 0.5$.

The *nbRep* value helps a peer to evaluate the forwarding behaviour of its neighbors. Neighbor nodes that exhibit no-forwarding or free-riding behaviour will have their *nbRep* score decreased. Peer i with the above behaviour will earn smaller slices of the node's capacity C_i as per Eq. 2. When a neighbor i receives low $nbRep_i$ evaluation, its chances of being rejected by our *topology adaptation* algorithm increase. A node may attack our protocol by forwarding many counterfeit results for non-existing content in order to improve its *nbRep* evaluation. This will increase its reputation evaluation and its probability of being selected as a resource provider. To protect itself from this behaviour, a peer uses recommendations from third parties as detailed in Section 3.2.4. Also, if the dishonest node is selected as a provider and the peer downloads content from it, the dishonest node will be punished by the *proRep* evaluation parameter once the content is discovered to be fraudulent, as explained in the following section.

3.2.3 Computing the proRep parameter

Each time the peer downloads content from a *collateral* or *neighbor* node i, the application asks the user to evaluate the downloaded content. This triggers the evaluation of node i as content provider ($0 \leq proRep_i(t) \leq 1$) and is calculated as follows. If the content quality is acceptable, the user gives a *good* rating so the $proRep_i$ evaluation of the provider node is increased as: $proRep_i(t) \leftarrow proRep_i(t-1) + \delta$. The user may also rate the content as *bad* in case it is of poor quality, or *dangerous* if it contains malware. As a result, the provider node $proRep_i$ evaluation that was rated as *bad* is decreased as: $proRep_i(t) \leftarrow proRep_i(t-1) - \lambda\delta$. If it was rated as *dangerous*, the evaluation is decreased as: $proRep_i(t) \leftarrow proRep_i(t-1) - \theta\delta$. In both cases, $\theta > \lambda > 1$. Through extensive simulation experiments we found that viable values for these parameters are $\delta = 0.2$, $\lambda = 2.5$ and $\theta = 5$. In practice, the individual values of these parameters are not of significant importance to our reputation calculation. What is important is the relative magnitude of the values, which determines the severity by which we wish to punish misbehaving peers. If the $proRep_i(t)$ drops below zero, its value is set to zero and, if it elevates above one, its value is set to one.

Finally, after each download, a file integrity check is performed using the already known MD5 file digest. If the calculated digest does not match the previously known digest, the downloaded resource is discarded. There are two alternatives to further strengthening the management of content through the MD5 digest. If the content in question is available from multiple peers but the MD5 values differ among peers, a majority vote can be used to determine the authentic content. This is a best-effort mechanism and is vulnerable to adversaries with sufficient resources. As previously discussed, a Byzantine protocol [5] could be implemented to concretely verify the authenticity of information. This, however, is beyond the scope of this paper and is left for future implementation.

3.2.4 Computing the othersRep parameter

Every time the peer has to select from a set of content-provider nodes (these could be collateral nodes or

direct neighbors), it requests from each one a list of nodes that shall act as its recommenders. To this effect, the peer sends to the potential provider node i a *GiveRecomm* message and the node responds with a *TakeRecomm* message. In order to be selected, node i has to furnish a set of recommenders. In general, the number of the recommenders is a fraction of node i's neighbors. Our rationale is that a malicious node will be surrounded by less honest neighbors than an honest one (based on the operation of the topology adaptation as detailed in Section 3.3). Hence, a malicious node cannot easily provide a set of good recommenders.

Once the list of recommenders is received, the peer proceeds by asking its trusted neighbors and a fraction of the recommenders to give their evaluations of node i. This is done with an *EvalNode* message, which carries the peer's public key. Subsequently, the peer collects all encrypted *EvalNodeReply* responses containing the evaluations. The messages are decrypted and the tampered ones are discarded. The peer then computes the average score of evaluations for the candidate provider node i as:

$$avgRecScore_i = \frac{\sum_{r=1}^{l} \alpha_r \cdot recScore_r}{l} \tag{5}$$

where $recScore_r$ is the evaluation received from node r and $r = 1,\ldots,l$. A dishonest node may control a number of other nodes and use them to send high evaluations of itself. In order to confront this type of attack, we use the α_r factor. It represents the trustworthiness of each recommender. This allows the peer to give more importance to evaluations from already trusted neighbors, than to those that came from unknown sources. For trusted nodes, the α_r value is set to their Rep_i value. For unknown nodes, it is set to $Rep/2$ of the least trusted node (i.e. the one with the minimum Rep_i value). Finally, the $0 \leq othersRep_i(t) \leq 1$ evaluation of a candidate-provider node i is also epoch-dependent and has memory depth (controlled by $\mu \in [0, 1]$), like $nbRep_i$, and is computed as:

$$othersRep_i(t) = \mu \cdot othersRep_i(t-1) + (1-\mu) \cdot avgRecScore_i \tag{6}$$

3.2.5 A concrete example

In order to better understand how peer reputation affects the allocation of service provisions, consider the following example. Let's assume that node j provides $C_j = 50$ kbps of its total network bandwidth in order to provide service to other peers. Also let's assume that node j has only 3 neighbor nodes (n_1,n_2,n_3) with respective reputations $nbRep_1 = 0.5$, $nbRep_2 = 0.7$, and $nbRep_3 = 0.2$. For the denominator in Eq. 2 we have $\sum_{k=1}^{3} nbRep_k = 1.4$. Therefore,

n_1 will get $c_1 = 50 \cdot nbRep_1/1.4 = \lceil 17.8 \rceil = 18$ kbps,
n_2 will get $c_2 = 50 \cdot nbRep_2/1.4 = \lceil 25 \rceil = 25$ kbps,
n_3 will get $c_3 = 50 \cdot nbRep_3/1.4 = \lceil 7.1 \rceil = 7$ kbps.

Suppose now that n_1 wants to receive better service and to increase the 18 kbps that node$_j$ has allocated for n_1 to 20 kbps. If the reputations of the other neighbors remain the same, from Eq. 2 we have:

$$20 = \frac{50 \cdot nbRep'_1}{0.7 + 0.2 + nbRep'_1} \Rightarrow nbRep'_1 = 0.6$$

Therefore, to get the additional 2 kbps from node$_j$, n_1 has to increase its $nbRep$ by 0.1. From Eq. 4 we have

$$nbRep'_1 = \mu \cdot nbRep_1(t-1) + (1-\mu) \cdot \frac{R_1}{totalR}$$

$$0.6 = \mu \cdot 0.5 + (1-\mu) \cdot \frac{R_1}{totalR}$$

$$R_1 = \frac{(0.6 - \mu \cdot 0.5)}{(1-\mu)} \cdot totalR$$

With $\mu = 0.5$ we get $R_1 = 0.7 \cdot totalR$. Therefore, n_1 must return at least 70% of the responses that node$_j$ will receive for its new query. In general:

$$R_k = \frac{(nbRep'_k - \mu \cdot nbRep_k)}{(1-\mu)} \cdot totalR$$

3.3 The *topology adaptation* module

Peers with similar content and effective cooperation are linked together by the *topology adaptation* module. Its main objective is to augment the overlay network by altering the direct connections between the peers. In this regard, the *topology adaptation* establishes direct connections between a requesting peer and its discovered counterparts. For this purpose, it uses the evaluation of the *reputation management* module.

While using the P2P network, the peer calculates the reputation of both neighbor and collateral nodes. Using Eq. 2, the peer can evaluate the services provided by its neighbors. The first step is for the peer to compute its satisfaction from the current formation of the overlay network. Every node features a specific bandwidth capacity that constraints its maximum number of connections with neighbors. Exceeding this number will

result in node overloading which in turn leads to failing requests and degrades the service quality. To prevent this, an *AGP* peer defines a minimum allocation capacity (denoted as *minAllocation*) for each connected node. The maximum number of linked nodes is:

$$maxNumberOfNeighbors \leq \frac{Capacity}{minAllocation} \quad (7)$$

Another important parameter is the *SatisfactionThreshold*$\in [0, 1]$ which defines a limit over which a peer is "satisfied" with its current P2P connections. This number also contributes to how often the topology adaptation procedure is triggered. A *SatisfactionThreshold* closer to one indicates a node keen to search for and adapt to a better neighborhood. Conversely, a value closer to zero indicates a peer satisfied with its placement in the network.

If the peer has less neighbors than those defined by *maxNumberOfNeighbors*, its satisfaction is set to zero in order to force the peer to rapidly make more direct connections with collateral nodes. Otherwise, the peer's satisfaction is set to the average total reputation score of its neighbors according to the following equation:

$$satisfaction(t) = \frac{\sum_{\forall i \in neighbors} Rep_i(t)}{numberOfNeighbors} \quad (8)$$

Algorithm 2, named *Adaptation check*, describes the topology adaptation process. The node maintains a desired satisfaction level (as defined by the *SatisfactionThreshold*). This parameter is defined by the user and can be changed dynamically. The first step is to compute the current satisfaction of the peer as shown previously and then compare it with the *SatisfactionThreshold*. There are two cases: a) the node is satisfied with its current topology, and b) the node is dissatisfied. In case a) the node is already satisfied and there is nothing that we have to do. If case b) occurs, then the node has the opportunity to do two things. First, if it has not reached its *maxNumberOfNeighbors*, it can connect with one more discovered peer with high reputation. If the node has used all its available links, it attempts to discover more neighbors by invoking the *Topology Adaptation Algorithm* described in Algorithm 3, where the function departNeighbor() helps find candidate nodes to be evicted. In both cases, the node exchanges content description messages with a number of candidate peers that have high reputation. These messages provide the node with the content descriptions of the candidates, their operational state and satisfaction level. The interesting case is when the node has reached its *maxNumberOfNeighbors*. Then, the node compares its content similarity with that of its neighbor and the new collateral nodes. A new link is made only if a collateral node has higher similarity with the node. After that, the cache and the network links are properly updated.

Algorithm 2 *Adaptation check Algorithm*

```
1: if (satisfaction < SatisfactionThreshold) then
2:    select node_i ∈ CollateralNodes with max{Rep_i}
3:    send RequestContent message to node_i
4:    wait until receive SendContent message from node_i
5:    if (node_i is not satisfied and has not reached maxNumberOfNeighbors) then
6:       TopologyAdaptationAlgorithm(node_i)
7:    end if
8: end if
```

3.4 *AGP* searching mechanism

When a node initiates a query without having prior content and reputation information for its neighbors, it carries out random walks to reduce query overhead. If, however, there is meta-data in the node's cache (see Section 3.5), our search algorithm will carry out a search using more efficient paths. Hence, our search uses an augmented breadth first search, which is directed only to neighbors with good reputation evaluation or ones that are known to contain the requested content. The node selects peers with high reputation evaluation due to their provision of good service(s) and, thus, avoids nodes with no-forwarding behaviour, free-riders and malicious nodes.

Algorithm 4 outlines how our search mechanism uses the following user-set key parameters: *1) MaxResponses*: indicates the maximum number of responses that an initiator peer requests, *2) RepThreshold*:

Algorithm 3 *Topology adaptation Algorithm*

```
1: input: collateral node
2: if (number of node's neighbors + 1 < maxNumberOfNeighbors) then
3:    create new direct link with collateral
4:    send SendContent message to new neighbor
5: else
6:    D ← departNeighbor(neighbor's peers)
7:    if (similarly(this node, collateral node) > similarity(this node, D) and (D is not empty) then
8:       send Drop message to D
9:       create new direct link with collateral
10:      send SendContent message to new neighbor
11:   else
12:      REJECT new connection with collateral node
13:   end if
14: end if
```

Algorithm 4 *Search Algorithm*

```
1: FreeSet ← { i: ∀ i ∈ neighbor_nodes and i is not overloaded }
2: RepSet ← { i: ∀ i ∈ FreeSet and Rep_i(t) ≥ reputationThreshold }
3: ContentSet ← { i: ∀ i ∈ FreeSet and i has the requested content }
4: ContentRepSet ← { i: ∀ i ∈ RepSet and ContentSet }
5: if ( ContentRepSet ∉ ∅ ) then
6:    for (each j ∈ subset of ContentRepSet) do
7:       forward query to j with reduced maxResponses
8:    end for
9: else if ( RepSet ∉ ∅ ) then
10:   for (each j ∈ subset of RepSet ) do
11:      forward query to j with maxResponses
12:   end for
13: else if (ContentSet ∉ ∅ ) then
14:   for (each j ∈ subset of ContentSet) do
15:      forward query to j with reduced maxResponses
16:   end for
17: else
18:   for (each j ∈ subset of FreeSet ) do
19:      forward query to j with maxResponses
20:   end for
21: end if
```

designates the reputation threshold over which a node considers a neighbor to be a useful provider.

The node first computes the *FreeSet* that contains the non-overloaded neighbors. When a node is overloaded, it informs its neighbors with an *Overloaded* message. When it returns to a normal operational state, it sends a *Free* message. This way, the operational states of the neighbors are consistently updated. Using the *FreeSet*, the node calculates the *RepSet* that contains the non-overloaded neighbors having reputation values over the *RepThreshold*. Furthermore, the node uses *FreeSet* to calculate the *ContentSet*. The latter contains the non-overloaded neighbors which maintain the requested content. A final set, the *ContentRepSet*, is computed as the intersection of the *RepSet* and the *ContentSet*.

If the *ContentRepSet* is not empty, the node reduces the maximum number of results and sends the query to a percentage of the *ContentRepSet* peers. Otherwise (i.e., *ContentRepSet* = 0), the query is randomly sent to a percentage of peers that belong to *RepSet*. In this manner, the node selects paths containing peers with good reputation score, even if they do not feature peers with the requested content. In other words, the node avoids paths with peers that do not provide forwarding services and offer low-grade or malicious content. In case the *RepSet* is also empty, the node checks the *ContentSet*. If the *ContentSet* is not empty, the node reduces the maximum number of results and sends the query to a percentage of the *ContentSet* peers. As the node which initiates the query maintains updated content descriptions about neighbor resources, the neighbors with the requested content are known a-priori. Should the *ContentSet* be empty, in order to reduce message overhead, the algorithm operates using a random walk. Thus, the query is sent to a fraction of its neighbors in order to avoid flooding the network with the same query.

3.5 *Cache module* components

The *cache module* has a number of specialized caches that maintain critical information for the operation of the *AGP*-protocol. These caches are:

- peer-self information cache (psi Cache): it maintains the standard configuration elements of the node including its unique *ID*, *maxNumberOfNeighbors*, the node's *public* and *private* keys, as well as the *Satisfaction Threshold*.
- known peer general information cache (kpgi Cache): it holds information about the discovered nodes. The kpgi Cache stores the *ID*s and the following parameters about each discovered node:
  - three key reputation parameters which are used for reputation evaluation, namely, *nbRep, proRep* and *othersRep*. These parameters indicate how the node evaluates: a) a peer as neighbor for its provided forwarding services (*nbRep*), b) a peer as content provider using the evaluation of its upload services (*proRep*), and c) a peer as provider using recommendations from third parties for its overall services (*othersRep*),
  - the current operational state of the node (i.e., whether it is overloaded).

  As the above information is heavily used by the *reputation* and *adaptation* components, each node maintains it in memory, using two hash-tables (as shown in Fig. 7): the first pertains to the *neighbors* and the second to the *collateral* nodes. Both hash-tables use as keys the node-*ID*s. If the node decides to suspend its operation for a period of time, it may elect to store its hash-tables to a file for later (re)use.
- resource general information cache (rgi Cache): it maintains information about the content stored in neighbors (as shown in Table 3). This portion of the cache contains: *XML* descriptions of the neighbors' content, their *MD5* digests, and the *ID*s of the nodes that this content originates from (along with their quality assessment if downloaded). This information is normally stored in a database table allowing for quick access to specific peer information.

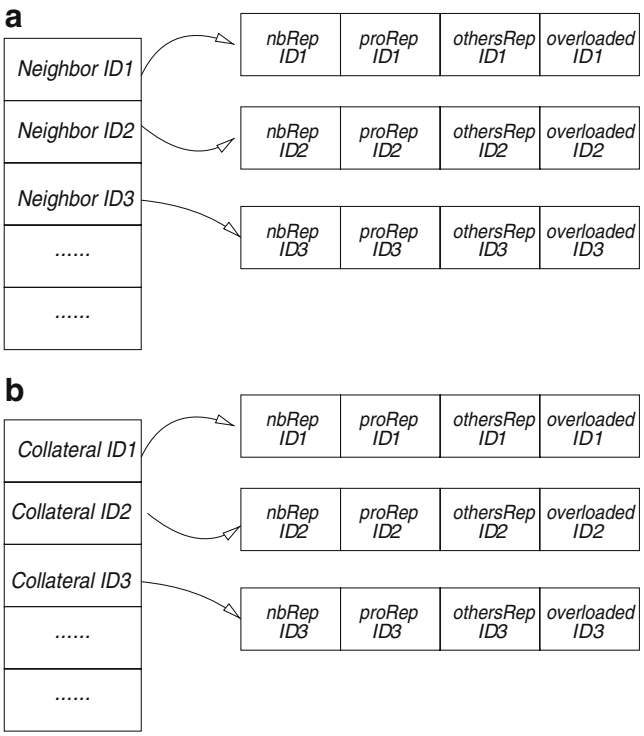


Fig. 7 kpgi Cache data structures (**a**, **b**)

- providers Evaluation cache (pEval Cache): stores recommendations regarding provider nodes offered voluntarily by third parties and for which the requesting peer might be interested in downloading from. Such recommendations are used for the computation of the *othersRep* parameter.
- query Forwarded cache (qForwarded Cache): stores the *IDs* of the nodes that each query was forwarded to. In this manner, a specific query is not forwarded to the same neighbor more than once and the node knows when the query has been sent to all of its neighbors. This information is stored as hash-table entries using as key the *GUID* of the query and value the neighbors *IDs* in the form $GUID$–{ $nodeID_1$,$nodeID_2$,...}. The qForwarded Cache is *LRU*-maintained, giving higher priority to more recent queries.

Table 3 rgi Cache mandatory fields; optional fields include *quality evaluation*

		Mandatory fields	
$Neighbor_1$	$File1_1$	XML description	MD5 digest
$Neighbor_1$	$File2_1$	XML description	MD5 digest
$Neighbor_2$	$File1_2$	XML description	MD5 digest
...	...	...	...
$Neighbor_N$	$File1_N$	XML description	MD5 digest

4 Empirical results

In this section we assess the performance of the proposed protocol. We evaluate not only the reputation and adaptation scheme but also our search mechanism. The successful operation of the *AGP* protocol depends on the parameters that each node uses to determine its connectivity with the other peers. These parameters are listed in Table 2. We show that the protocol successfully discovers and punishes peers with dishonest behaviour. We also show that our search mechanism exhibits better overhead than previous proposals based on *Gnutella*.

4.1 Node behaviour

The network consists mainly of honest nodes that contribute their resources. The honest nodes provide fine quality files and query forwarding service. In the network, there also exists a population of dishonest nodes that do not operate like the honest ones. We have defined and simulated four types of dishonest node behaviours. The first type consists of *malicious* nodes that constantly upload files with malicious content such as virus infected files. Nevertheless, these nodes always present good forwarding services and propagate the receiving queries from other nodes to the network. The second type of dishonest nodes is *free-riders* that refuse to share their content. The third type consists of nodes that exhibit *dynamic dishonest* behaviour. These nodes behave as honest ones for a period of time and then for the remainder of their lifetime behave randomly either as malicious/free-riders or as honest nodes. The fourth type are dishonest nodes that provide low quality content.

4.2 Content distribution

In our simulations, the shared items are files belonging to a number of content *categories*. To define how to distribute distinct files to the peers, we use the measurements provided by [26], as well as those by [9], where 72% of the total nodes will have a file that belongs to 30 or less categories, 10% of the total nodes will have a file that belongs to at most 50 categories, and 18% of the total nodes will have a file that belongs in one category. Each peer supports a set of categories. The assignment of items among the network nodes is based on the distribution in [9] over 200 content categories. Based on the content categories that it supports, each node is associated with a number of files in these categories. This is achieved using a uniform random distribution. Any connected peer that is not

overloaded can issue new queries. We model the time that the peer is connected based on the cumulative distribution function in [26]. However, we used epochs instead of minutes to indicate the passage of time. This distribution also defines our churn rate since we assume that, once disconnected, a peer has left the network. At a later time, the peer may re-enter a different part of the network. In our simulation model the peers can execute exact or keyword related queries.

4.3 Simulation execution

We have used the *PeerSim* environment [15] to simulate the operation of *AGP* and to carry out objective comparisons with the *Gnutella* v0.6 protocol. The *PeerSim* engine is epoch-based. During an epoch, each node can be up (connected), down (disconnected) or overloaded. If a node is up, it can submit new queries, wait for incoming responses, select from a pool of possible providers that responded, evaluate other peers and download a file. The last three steps are repeated until a peer downloads a file or there exist no further responses from peers with high reputation. Furthermore, the nodes can invoke their adaptation module in order to enhance their location in the P2P network.

Our simulation parameters are listed in Tables 2 and 4. Each of our experiments lasts for 1000 epochs. First, we evaluate the reputation and adaptation schemes of our proposed protocol. A fundamental decision deals with the calibration of Eq. 2. Based on a wide range of simulations, we have concluded that the best performance can be obtained with the values $c_1 = 0.5$, $c_2 = 0.4$ and $c_3 = 0.1$. We have concluded that these values represent a viable configuration which allows peers to make good overall evaluations of their neighbors. Additionally, this configuration gives equal role to message forwarding (c_1) and to download provision ($c_2 + c_3$). To ascertain that nodes take advantage of the adaptation mechanism in an equitable manner across the network, each peer activates the adaptation module every 20 epochs, except in the case of the search evaluation where the adaptation module remains dormant.

Furthermore, we compare the *AGP* search algorithm with the *Gnutella DQP (Dynamic Query Protocol)*. Because only the ultra-peers[2] are responsible for searching the *Gnutella* network, we only consider the ultra-peer layer overlay in our simulations. This ultra-peer-layer forms a stable core-layer, whose nodes are connected randomly with an average connectivity degree of 30 [29]. So we approximate the stable core of *Gnutella* topology through a Random Graph with an average degree of connectivity $k = 30$.

In general we report results derived from a *Random Graph* network topology. When we evaluate the reputation and adaptation schemes as well as the search mechanism, we define the connectivity degree $k = 30$.

During the evaluation of the reputation and adaptation schemes, the nodes submit keyword related queries. The nodes are not interested in exact file matches, rather, they search for files with content descriptors that contain a specific keyword. Therefore, one or more distinct files can satisfy their query. On the other hand for the evaluation of the search algorithm we inject a specific file into a percentage (1%) of the total node population, in order to force peers to search for scarcely replicated items. In the latter scenario, we evaluate only exact match queries.

4.4 Protocol evaluation

The main objective of our evaluation is to establish the improvement of the overlay network topology through adaptation based on both content quality and reputation. This improvement leads to reduced overhead when using the proposed search algorithm.

In this context, we pursue four specific goals: *1)* we examine whether our reputation management can appropriately evaluate nodes for their behaviour. We investigate the effectiveness of our topology adaptation module when it comes to confronting dishonest peers. *2)* We seek to ascertain that the neighborhood for honest nodes can improve over time. The content similarity metric helps measure this, as it should increase for peers that successfully reposition themselves in better neighborhoods even in the presence of a large population of dishonest nodes. *3)* We demonstrate that the overall P2P network efficiency improves significantly. Metrics that can help us in this direction are the decreasing ratio of malicious vs. high-grade downloads and the number of malicious responses. *4)* We examine if our proposed search algorithm is more efficient than *Gnutella*'s *DQP*. The metrics we use are the message overhead, the send/receive message ratio and the number of received responses.

4.4.1 Evaluation of node behaviour

Figure 8a shows how the nodes that present five different types of behaviour get separated over time with the help of our *Reputation Management* module. The figure

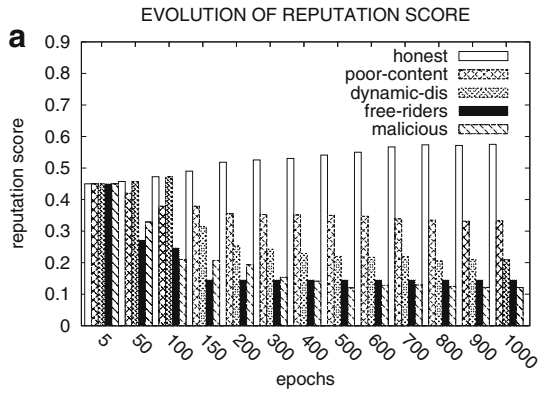
[2]Ultra-peers are nodes with high bandwidth capacity and high number of direct connections.

Table 4 Simulation settings: networks size, content distribution and peer behaviour

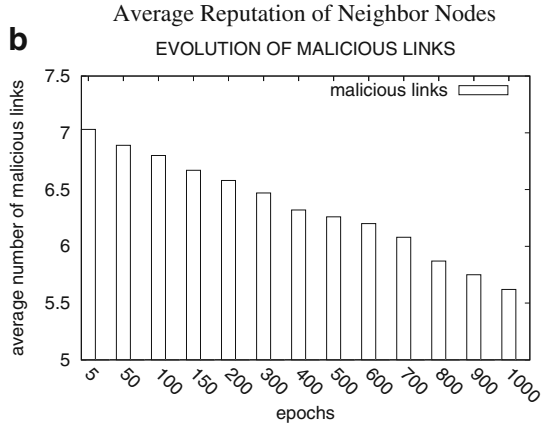
Network	Maximum # of peers	10,000
	# of honest peers	800 or 600
	# of dishonest peers	200 or 400
	# of initial neighbors of honest peers	3
	# of initial neighbors of dishonest peers	5
	Maximum # of allowed connections for reputation and adaptation evaluation	30
	Average # of allowed connections for search evaluation	30
	# time to live for query messages	5
Content distribution	# of possible distinct files in the network	1000
	# of distinct files at peer *i*	File distribution in [26]
	Set of content categories supported by peer *i*	Distribution in [9] over 200 content categories
	# of distinct files at peer *i* in category *j*	Distribution in [9] over peer *i*'s
	% of time peer *i* is up and processing queries	Distribution in [26] over [0%, 100%]
	% of injected special files used for search evaluation	1% of total nodes
Query mechanism	Evaluation of reputation and adaptation scheme	Use of keyword based queries
	Evaluation of search mechanism	Use exact match queries
Peer storage	Maximum number of messages in peer's *i* queue buffer	50
	Number of free slots in queue left in order to considered overloaded	5
Honest peer behaviour	% of download requests in which honest peer *i* returns authentic file	90%
	% of download requests in which honest peer *i* fails to evaluate a corrupted file *i*	5%
	Forwarding queries	Always except if overloaded%
Malicious peer behaviour	% of download requests in which malicious peer *i* returns inauthentic file	80%
	Forwarding queries	Always except if overloaded%
Free-rider (including non-forwarder) peer behaviour	% of download requests in which free-rider peer *i* returns file	0%
	Probability peer *i* to forward queries	Uniform random distribution over [0%, 10%]
Dishonest peer with dynamic behaviour	Probability dishonest peer *i* to exhibit dynamic behaviour	Uniform random distribution over [0%, 50%]
	Number of turns until peer *i* starts to exhibit malicious behaviour	100
	% of download requests in which peer *i* behave dishonest and *i* returns inauthentic file	80%
	Probability peer *i* to forward queries	Uniform random distribution over [0%, 50%]
Simulation	# of simulation cycles in one experiment	1000
	# of cycles between invocation of adaptation module	20
	# of experiments over which results are averaged	5

depicts the average reputation of the honest nodes' peers. As peers evaluate the services received by their honest neighbors in a positive manner, the reputation of the latter increases. Free-riders are punished for not offering any services and their reputation scores rapidly decrease. The reputation scores of the malicious peers are also degraded and remain lower than the scores of their free-riding counterparts. This happens despite the fact that the malicious nodes provide useful forwarding services, and is due to the heavy punishment of the θ parameter described in Section 3.2.3. Nodes that provide low quality content (but not dangerous one) also receive low evaluation scores. These nodes, however, provide good forwarding services receiving higher scores than the malicious peers. Lastly, the nodes that exhibit dynamic dishonest behaviour are more difficult to evaluate. Nevertheless, over time, the reputation scheme succeeds in discovering them. This happens even if they have built a high reputation score in the first 100 epochs and then start their dynamic dishonest behaviour.

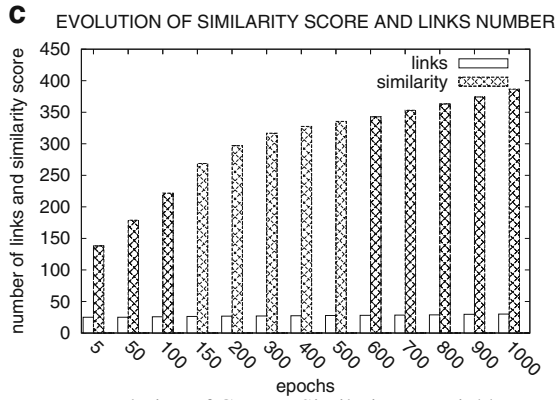
Clearly, the reaction of honest nodes to dishonest parties is not instant and takes a number of epochs with more time needed in the case of the malicious and dynamic dishonest peers. In general we conclude that the non-forwarding behaviour is easily discovered



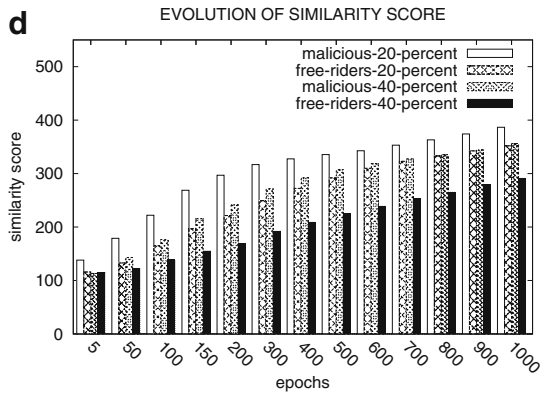
Average Reputation of Neighbor Nodes



Average Number of Links to Malicious Nodes



Evolution of Content Similarity vs.Neighbor Connections



Content Similarity Evolution in a P2P-network with Dishonest Parties in Various Distributions

◀**Fig. 8** Confrontation of malicious nodes and improvement of neighborhood overlay network topology (**a**–**d**)

and punished. The malicious behaviour needs more time and low quality downloads to be discovered. As expected, the dynamic behaviour is even more difficult to be discovered, but our reputation scheme achieves this goal. After a number of epochs, the honest nodes have a more comprehensive and accurate view of the network and its provided services.

Figure 8b shows that the average number of overlay connections between the honest and the malicious nodes drops over time. As we already showed, the honest nodes properly evaluate the malicious peers reputation. When the topology adaptation takes place and a node appears to maintain malicious neighbor peers with reduced reputation scores, the node drops its connections with malicious peers and tries to establish direct overlay links with honest nodes. The same principle that applies to the free-riders is also valid for the poor content providers as well as the dynamically dishonest peers.

4.4.2 Neighborhood overlay network topology

Figure 8c shows how the average content similarity among the nodes and their neighbor peers increases, even if the nodes have reached their maximum number of direct overlay link connections. As presented, the content similarity score increases not only initially, when the topology adaptation tries to connect the node with more peers in order to increase its satisfaction level, but continues to increase as the topology adaptation module swaps neighbors with collateral peers holding similar content. This last result is confirmed by Fig. 9a which shows that the improvement of the similarity score per new direct link continues to increase even after the initial 150 epochs have elapsed, and most nodes have reached their maximum number of neighbors. This is due to the adaptation algorithm: before the 150 epochs, most nodes simply accept new nodes for direct linkage without considering their similarity score. This happens in order to quickly reach the maximum number of neighbors and to increase their satisfaction level. After the initial 150 epochs, the majority of peers have reached this connectivity limit and the adaptation algorithm carries out a careful selection of collateral peers, that are candidates for becoming new neighbors, based on the content similarity. As Fig. 8d shows, the above result holds for a varying number of malicious and/or free-rider nodes.

4.4.3 Enhancing overall network efficiency

Figure 9b depicts how the average ratio of malicious over high-grade downloads develops over time. This ratio articulates the quality of downloads provided by the overall network. There is a substantial decrease in this ratio during the early stages of the experiment that reflects an enhanced network operation.

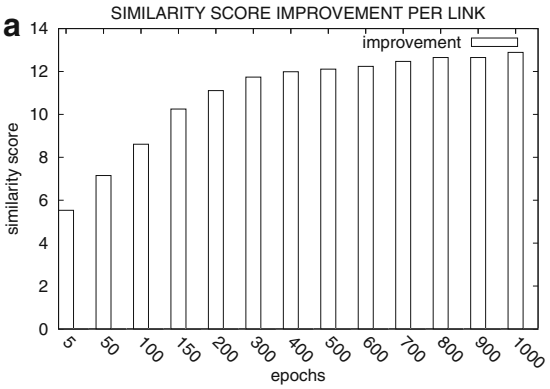
Furthermore, Fig. 9c presents the evolution of the average number of malicious downloads over the number of malicious message responses that the node received. The above ratio decreases and this result must be evaluated along with the outcomes of Fig. 9b. In combination, both indicate that the number of malicious downloads is reduced over time. Indeed, the results gathered after a number of conducted experiments and presented in Table 5 show a substantial decrease in the number of malicious downloads. Furthermore, they show that the ratio of fine to malicious responses increases significantly. In other words, as the time passes by the network operation improves, because the dishonest peers are removed and the honest ones receive less malicious responses to their queries.

4.4.4 Efficiency of AGP searching mechanism

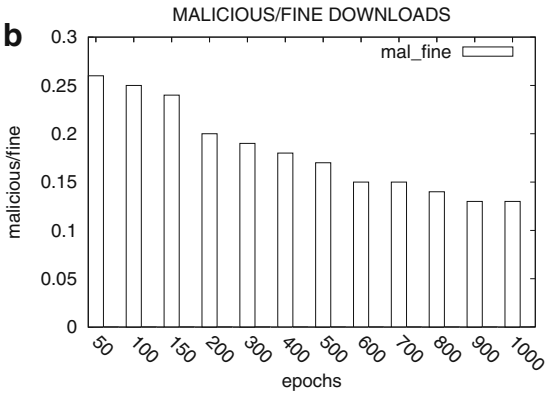
In order to evaluate the efficiency of the *AGP* search algorithm compared to *Gnutella DQP*, we use the following metrics:

- *Overhead* as expressed by the overall number of messages sent in the network.
- *Send/Receive Ratio*: this is the number of sent messages divided by the number of received query response messages. A low ratio implies a highly efficient query processing.
- *QueryHits* indicates the number of the received query response messages by the request initiator node.
- *Query Response Time* for the queries which were answered in the specific epoch.

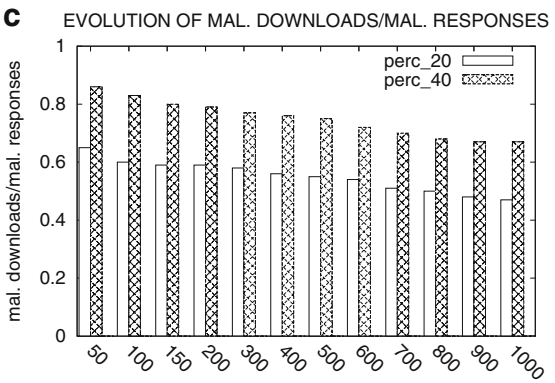
The main objective for evaluating the *AGP*-search algorithm is to see if it can exhibit the same capability in returning hits as the *Gnutella DQP* protocol but with improved network resource utilization. Figure 9d confirms that the *AGP*-search accomplishes better utilization of network resources. In this figure, the *AGP*-search needs significantly fewer messages as compared to *Gnutella*. This happens because it selects paths con-



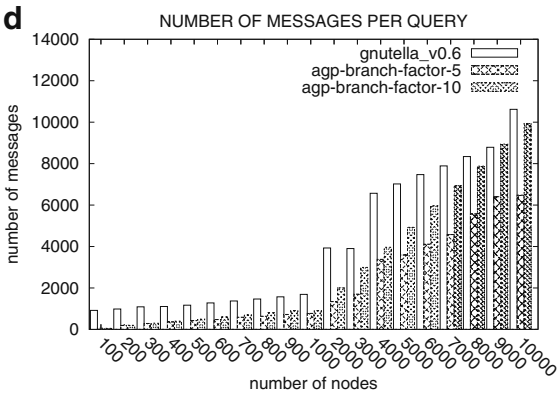
Increase in Similarity Score per New Direct Link.



Malicious/High-grade Downloads in a network with 40% Malicious nodes.



Number of Malicious Downloads/Number of Malicious Responses



Total Number of Send Messages per Query

Fig. 9 Improving neighborhood overlay network topology and overall network efficiency (**a**–**d**) ▶

Table 5 Simulation statistics on fine/malicious downloads and responses

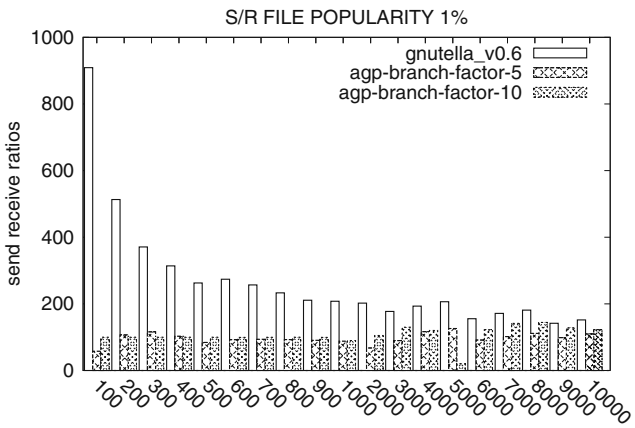
Malicious 20% of the overlay network nodes										
Epochs	0–100	100–200	200–300	300–400	400–500	500–600	600–700	700–800	800–900	900–1000
Malicious downloads	96	3	1	0	0	0	0	0	1	0
Fine downloads	1424	1667	1658	1696	1660	1672	1677	1685	1672	1642
Fine/malicious responses	5.22	5.94	6.3	6.2	6.33	6.38	6.52	6.46	6.46	6.53
Malicious 40% of the overlay network nodes										
Epochs	0–100	100–200	200–300	300–400	400–500	500–600	600–700	700–800	800–900	900–1000
Malicious downloads	230	12	4	1	4	9	9	0	5	1
Authentic fine downloads	1290	1658	1653	1691	1655					
Fine downloads						1663	1668	1685	1668	1641
Fine/malicious responses	2.39	2.99	3.18	3.16	3.16	3.14	3.15	3.27	3.31	3.33

taining peers either storing the requested content or having high reputation scores. Increased reputation means that the possibility is low that this peer is dishonest and may provide malicious content or exhibit free-riding behaviour. The *Gnutella*'s search algorithm selects paths that may contain free-riders and therefore stop forwarding the query. The *Gnutella* algorithm also fails to discriminate the honest from the malicious content providers.

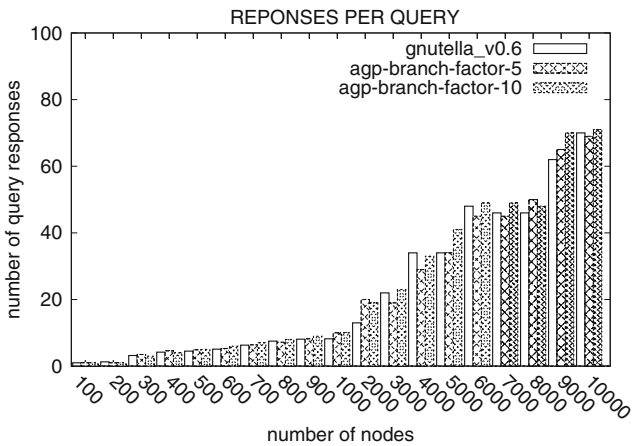
In Fig. 10b, *AGP*-search exhibits almost the same behaviour as the *Gnutella DQP*. This occurs because the *AGP* search uses all the available information about the neighbor nodes and forwards the queries to the nodes that possess the item in question. The *Gnutella* search algorithm uses an iterative two-step process to discover these nodes. These two different mechanisms effectively present almost the same capability in returning results. However, in terms of network utilization, *AGP* demonstrates much lower send/receive ratio than *Gnutella* as Fig. 10a depicts. We believe that these results will be further improved if the *AGP* search algorithm also operates in a two step process, similarly to *Gnutella DQP*.

4.4.5 Topology adaptation overhead

In order to evaluate the messaging overhead due to the *Topology Adaptation* Module, we perform an experiment where we vary the *SatisfactionThreshold*. Furthermore all nodes are considered to be honest and have an initial reputation of 0.45. We run experiments with two satisfaction threshold values: 0.5 and 1.0; the former is close to the average reputation of the nodes (remember that a node's satisfaction is the average score of its neighbors reputation), while the latter is the maximum value. High values of the *SatisfactionThreshold* result in peers constantly seeking improvements in their neighborhood topologies. As shown in Fig. 11, with a *SatisfactionThreshold* of 1.0, the topology adaptation overhead is significantly higher and consistently increases over time compared to the case where *SatisfactionThreshold* is set to 0.5. Even if our protocol is designed to limit the extensive removals (a neighbor with less that 3 direct connections is never dropped) in order to avoid creating a disconnected graph, a high satisfaction threshold limits the scalability of the network.



(a) Send/Receive Ratios



(b) Number of Query Responses delivered

Fig. 10 Evaluation of *Gnutella* and *AGP* search algorithms in a Random Graph network with $k = 30$ (**a**, **b**)

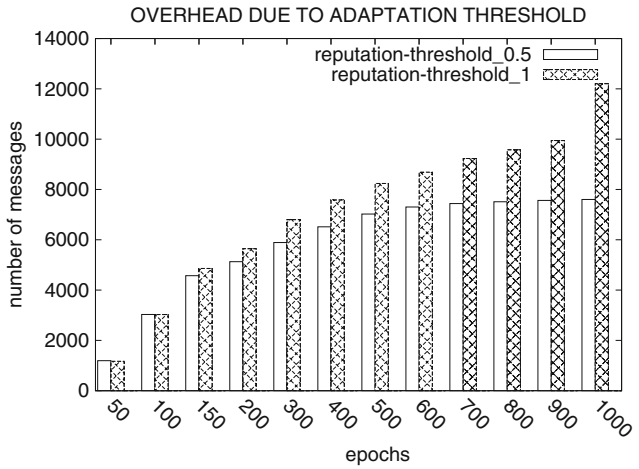


Fig. 11 Evaluation of *AGP* topology adaptation overhead

4.4.6 Handling whitewashing attacks

An effective measure against whitewashing attacks—where malicious nodes periodically reconnect with newly generated IDs in order to "wash" themselves off accumulated bad behaviour—is to give newcomers low initial reputation scores. Essentially this makes it harder for new nodes to earn the trust of their peers, but also makes it harder for malicious nodes to join new neighborhoods. As shown in Fig. 12, in this experiment, initially all peers start with the same reputation score. At epoch 50, whitewashing nodes exit the network and re-join under new identities. We can see that the lower the default reputation scores are, the more difficult it is for the whitewashing nodes to re-gain trust and their reputation remains low throughout the experiment. When starting with a reputation value of 0.45, peers can more quickly regain their neighbors' trust. In contrast, with values below 0.45, as whitewashing nodes re-join the network their reputations remain low, making it more difficult for them to join honest neighborhoods and to spread malicious files.

4.4.7 Query response time

We performed extensive experiments in order to examine the response time of the *AGP* versus the *Gnutella* protocol v0.6. As detailed in Section 3.4, unlike *Gnutella*, *AGP* uses content information to route queries. Moreover, the adaptation mechanism of *AGP* creates a topology that better satisfies each node. We report the query response time for the searches submitted by a random peer in the network. To achieve a more realistic comparison with *Gnutella*, we freeze the *AGP* topology adaptation for the duration of the query executions. As shown in Fig. 13, for a particular query, *AGP* returns most of the results in the first 5 epochs, while the bulk of the search results arrive in epochs 50–60 when Gnutella DQP is used. Clearly, *AGP* executes queries more efficiently and delivers a better user experience compared to Gnutella. Furthermore, *AGP*'s superb query response time is achieved with much lower messaging overhead demonstrated in Section 4.4.4 as queries are mainly propagated among reputed peers and results are typically retrieved from neighbor nodes with authentic content.

5 Related work

The first significant aspect of our proposal is its reputation scheme. Several years of research on reputation

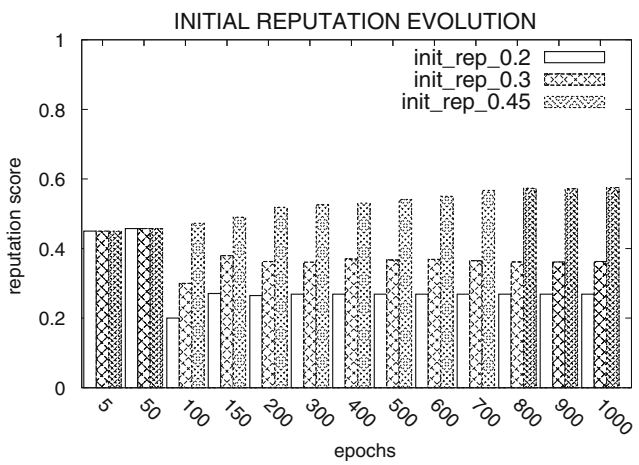


Fig. 12 Evaluation of *AGP* initial reputation scores and effect on whitewashing nodes

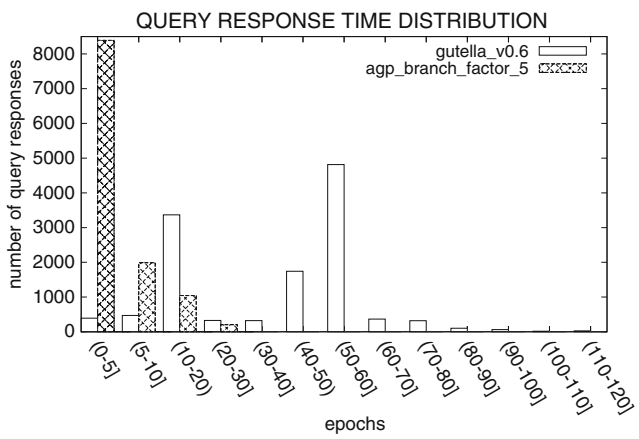


Fig. 13 Query responses as delivered throughout the experiment: *AGP* discovers the results for a query much sooner than *Gnutella*

algorithms has resulted in a number of interesting and valuable approaches. As we have shown in this paper, our solution differs from the previous proposals in a number of ways.

Various approaches exist that calculate a user participation level [4] and reward peers with high participation by giving higher priority to their requests, or by reducing their waiting times in the transfer queues. A more abstract approach is presented in [23], where utility functions based on the amount and popularity of content stored by the peer, are used to estimate a metric for their usefulness. Such mechanisms are generally easy to subvert by malicious users and make no consideration about the possibility of attacks.

In [27], the network relies on a centralized server to assign unforgeable identities, but reputation adjustment is handled among peers. This requires that both parties in a transaction sign a transaction certificate that is then presented when reporting on the outcome of the transaction. This method cannot prevent adversaries from submitting fraudulent reports about peers they have not interacted with in order to lower their reputation. As we have demonstrated, our evaluation scheme uses weights, making it less sensitive to this type of attacks because the node's own experience has the largest weight.

In [16], the Eigen-Trust algorithm is proposed, which produces global reputation ratings for users based on their history of uploads. These ratings can then be used by other peers to decide where to download files from. The global reputation values are computed from the local reputation values assigned to a peer by other peers, weighted by the global reputation of the assigning peers. This approach was found to reduce the number of malicious files on the network. This proposal is decentralized but is vulnerable to whitewashing and to colluding peers. The authors suggest that each agent separately weighs a globally computed rating with the personal opinions of trusted peers, when available. Our proposal uses a similar metric based on the weighted opinions of the trusted peers. Furthermore, our metrics are not only based on the history of uploads but also on the forwarding services provided by the peers. In a network with a high churn rate, we believe that methods that create global reputation ratings are not useful as they produce unnecessary overhead.

A partially centralized mechanism using reputation computation agents and data encryption is presented in [13]. The reputation values are calculated, encrypted, and stored locally using a reputation computation agent. Two different schemes for calculating reputation values are proposed: a credit/debit scheme, and a credit only scheme. The similarity with our protocol is that both give incentives to peers for cooperation. Also, both protocols take into consideration the peer's hardware and network abilities. On the other hand, this protocol is designed to give incentives for cooperation and does not include provisions for fighting peers with malicious behaviour.

A feedback-based reputation mechanism is presented in [31], where a scalar trust value for a peer is computed based on three factors: (1) the amount of satisfaction received by other peers in the system, (2) the total number of interactions, and (3) a balancing factor to offset the impact of malicious peers that misreport other peers' service evaluations. The reputation information is distributed in the network so that each peer maintains a portion of the total trust data. Though this reputation storage scheme differs from other reputation systems, it still requires cooperation from the peers for storing the reputations. Malicious behaviour is countered by having multiple peers responsible for storing the same database. Voting can be used if these databases differ. Trust is computed on the fly by querying multiple databases over the network. The distributed storage of information is a robust way of improving the quality of reported peer reputation. Similarly to this method, we query the community for reputation scores on a particular peer. In addition, we include a number of other factors, such as the local node's past experience, in producing a weighted score for a particular peer's reputation.

To a large extent P2P networks mimic the interactions of social groups of people. This idea is extended in [18], where a P2P algorithm for resource discovery is used by representing peers as people and connections as relationships. Such organization significantly reduces the overheads of obtaining information from neighbors. Similarly to [18], our proposed protocol attempts to create new connections based on query results and past experiences.

The feedback-based schemes discussed so far do not provide any special mechanism to detect and punish peers which disseminate rogue (altered or infected) files. The first to do so was [22], where the concept of suspicious transactions is used to detect and punish malicious peers. The goal is to maximize the user satisfaction and decrease the sharing of corrupted files. The algorithm detects malicious peers that send rogue files and attempts to isolate them from the system. The metric used is trust based on the accuracy and the quality of the file received. This protocol is designed for partially centralized systems with ultra-peers. Our reputation proposal differs because it is designed for purely unstructured networks. In [22], certain services provided by ultra-peers are differentiated and treated

separately. Our proposal expands this idea for the single peer, and not only for forwarding the responding results but also for providing incentives for query propagation. Furthermore, our proposal attempts to balance the need for limited overhead as provided by a small number [20] of peers, and the chance to collect useful information about a specific peer, using further resources [8]. At first, the information integrity is verified using hash functions that compare the message digest with the downloaded content and discard the latter if it is modified. Secondly, the content quality is verified by the user. In this manner, our protocol evaluates not only the provided content quantity but also its quality.

The second significant aspect of our protocol is its adaptation mechanism. To a certain extent, the area of self-organizing P2P networks has been researched previously. Noteworthy protocols and mechanisms include *APT* [7], GES [32] and GIA [6].

The *APT* protocol forms overlays based on the concept of the interaction topology which is essentially a graph whose edges are defined by downloads. A link is created between nodes i and j if node i has downloaded content from node j more than k times. *APT* seeks to transform the overlay network topology closer to the interaction topology. The *APT* operation is based on two premises: firstly, peers use their past interaction history to determine which nodes are likely to provide future downloads and secondly, the peers directly connect to these nodes. *APT* essentially produces "small-world" networks of "like-minded" users. Such networks are sparsely connected and display a high cluster coefficient [19]. *APT* deals with malicious peers based on interaction topologies in which a peer P evaluates a node R exclusively as a resource provider. P does not take into account neighbor nodes that may not provide direct downloads, but they may be able to provide good (forwarding) services. In this context, our solution rewards peers that may not have the requested content but act as intermediate nodes, forwarding the queries and providing paths to nodes that store the requested content, resulting in improved query performance.

The main objective of *GIA* is to improve the network scalability by ensuring that high-capacity nodes are those mainly used for query transferring and at the same time the low-bandwidth nodes remain within short reach. *GIA*-nodes independently try to create connections with nodes that have high bandwidth and increase their connectivity degree offering improved query responses while maintaining robustness to node failures [6]. *GES* employs a distributed content-based approach that organizes nodes into *semantic* groups. Here, every peer is represented by a vector of attributes that characterizes the node's documents. Furthermore, peers periodically issue random walk queries to discover new nodes and perform semantic-addition/replacement of neighbor nodes. The objective of this re-organization is to enhance query outcomes by creating node clusters with "similar" interests.

GIA and *GES* strive to improve scalability, to create more efficient network organizations, semantic groups and to return increased query results without taking into consideration malicious and/or low-grade peers that appear to proliferate into networks. Our protocol's key objective is not only to pursue peers' individual performance indicators but also to improve system-wide peer connectivity while isolating and ultimately downgrading the role of malicious peers.

6 Conclusions

In this paper, we have introduced methods for improving the network efficiency of the P2P *Gnutella*-like file-sharing protocols. Our proposal relies on reputation and topology adaptation schemes to improve the overall efficiency of the overlay network topology. The peers discard neighbors that exhibit malicious or selfish behaviour and create connections with nodes that provide good forwarding service, have similar content and provide high-grade resources. We show that, over time, the content similarity degree of the peers increases, the number of the malicious responses drops and so does the message overhead. Furthermore, we use a search algorithm that exploits not only the available neighbor content information, but also the reputation evaluations in order to select paths with nodes that provide good forwarding services. We show that our protocol is far more efficient than the *Gnutella DQP* algorithm as it returns the same number of results with much lower message overhead.

Our protocol, termed *Adaptive Gnutella Protocol* (*AGP*), differs from existing techniques as it also takes into consideration the forwarding service provided by the peers. In addition, *AGP* is designed to operate on a network with high number of malicious and selfish nodes through the use of a reputation scheme which evaluates the peers' past behaviour, provides incentives for node cooperation, and uses recommendations from distant nodes in order to enhance the peer's view of the network. Results derived through extensive simulations demonstrate the effectiveness of the proposed protocol. We have shown that *AGP* can isolate malicious nodes as well as free-riders and pushes them to the edge of the network, even in the presence of "sophisticated" adversaries that alternate their behaviour between honest and dishonest. Furthermore, our protocol helps a

peer to rapidly improve the composition of its direct as well as its collateral nodes attaining faster and better query responses. In addition to these improvements over *Gnutella* v0.6, *AGP* attains very short query response times, delivering the bulk of the results orders of magnitude faster than *Gnutella*.

One of the goals through the proliferation of P2P systems has always been the preservation of the "freedom" and the anonymity of the users. Although *AGP* imposes certain restrictions based on monitoring and behavioural analysis, it does so without revealing the users' identities. Furthermore, all of the monitoring performed in *AGP* is based on information that peers collect simply from their interaction with other peers. In this sense, *AGP* adds behavioural analysis based on existing information, consequently improving the overall user experience. We believe that a network with complete "freedom" and poor user experience due to malicious nodes would simply collapse as dishonest nodes and free-riders would flourish, while honest peers would desist from joining the network. This would be akin to an anarchy in a real-world social network.

In the future, we plan to examine the issue of network heterogeneity in terms of both physical links and network interfaces, address how this heterogeneity might affect our protocol, and explore the evaluation of peer reputation on a per-category basis. In the spirit of [11], we foresee improvements in our protocol if current network loads are considered when topology adaptation is performed.

References

1. McClain L (2004) RIAA posting bad music files to deter illegal downloaders. The Dialy Texan
2. BBC NEWS (2004) Viruses turn to peer-to-peer nets. BBC NEWS
3. Gnutella (2007) Gnutella protocol specification. http://gnutella-specs.rakjar.de/index.php/Gnutella_Protocol_Specification
4. KazaA (2007) KazaA official site. http://www.kazaa.com/us/index.htm, August
5. Castro M, Druschel P, Ganesh AJ, Rowstron AIT, Wallach DS (2002) Secure routing for structured peer-to-peer overlay networks. In: OSDI, Boston, December 2002
6. Chawathe Y, Ratnasamy S, Breslau L, Lanham N, Shenker S (2003) Making Gnutella-like P2P systems scalable. In: Proceedings of the 2003 ACM conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03). ACM, Karlsruhe, pp 407–418
7. Condie T, Kamvar SD, Garcia-Molina H (2004) Adaptive peer-to-peer topologies. In: Proceedings of the 4th international conference on peer-to-peer computing (P2P'04). IEEE Computer Society, Washington, DC, pp 53–62
8. Cornelli F, Damiani E, De Capitani di Vimercati S, Paraboschi S, Samarati P (2002) Choosing reputable servents in a P2P network. In: WWW '02: proceedings of the 11th international conference on world wide web. ACM, New York, pp 376–386
9. Crespo A, Garcia-Molina H (2004) Semantic overlay networks for P2P systems. In: AP2PC, New York, July 2004, pp 1–13
10. Dwork C, Naor M, Sahai A (2004) Concurrent zero-knowledge. J ACM 51(6):851–898
11. Exarchakos G, Antonopoulos N (2007) Resource sharing architecture for cooperative heterogeneous P2P overlays. J Netw Syst Manag 15(3):311–334
12. Feige U, Shamir A (1990) Witness indistinguishable and witness hiding protocols. In: STOC, Baltimore, 13–17 May 1990, pp 416–426
13. Gupta M, Judge P, Ammar M (2003) A reputation system for peer-to-peer networks. In: NOSSDAV '03: proceedings of the 13th international workshop on Network and operating systems support for digital audio and video. ACM, New York, pp 144–152
14. Hughes D, Coulson G, Walkerdine J (2005) Free riding on Gnutella revisited: the bell tolls? IEEE Distrib Syst Online 6(6):1–13
15. Jelasity M, Montresor A, Paolo Jesi G, Voulgaris S (2005) PeerSim: a peer-to-peer simulator. http://peersim.sourceforge.net, December
16. Kamvar SD, Schlosser MT, Garcia-Molina H (2003) The Eigentrust algorithm for reputation management in P2P networks. In: WWW '03: proceedings of the 12th international conference on world wide web. ACM, New York, pp 640–651
17. Li J, Loo BT, Hellerstein JM, Kaashoek MF, Karger DR, Morris R (2003) On the feasibility of peer-to-peer web indexing and search. In: 2nd international workshop on peer-to-peer systems (IPTPS '03), Berkeley, 20–21 February 2003, pp 207–215
18. Liu L, Antonopoulos N, Mackin S (2007) Social peer-to-peer for resource discovery. In: PDP. Washington, DC, pp 459–466
19. Liu L, Mackin S, Antonopoulos N (2006) Small world architecture for peer-to-peer networks. In: Proceedings of the 2006 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology. IEEE Computer Society, Washington, DC, pp 451–454
20. Marti S, Garcia-Molina H (2004) Limited reputation sharing in P2P systems. In: EC '04: proceedings of the 5th ACM conference on electronic commerce. ACM, New York, pp 91–101
21. Marti S, Garcia-Molina H (2006) Taxonomy of trust: categorizing P2P reputation systems. Comput Netw 50(4):472–484
22. Mekouar L, Iraqi Y, Boutaba R (2006) Peer-to-peer's most wanted: malicious peers. Comput Netw 50(4):545–562
23. Ramaswamy L, Liu L (2003) Free riding: a new challenge to peer-to-peer file sharing systems. In: Proceedings of the 36th annual Hawaii international conference on system sciences, 2003, 6–9 January 2003, p 10
24. Rivest R (1992) RFC 1321 - MD5 message digest algorithm. http://tools.ietf.org/html/rfc1321
25. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
26. Saroiu S, Gummadi PK Gribble SD (2002) A measurement study of peer-to-peer file sharing systems. In: MMCN '02: proceedings of multimedia computing and networking, San Jose, January 2002
27. Singh A, Liu L (2003) TrustMe: anonymous management of trust relationships in decentralized P2P systems. In: P2P '03:

proceedings of the 3rd international conference on peer-to-peer computing. IEEE Computer Society, Washington, DC, p 142
28. Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H (2001) Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM conference. San Diego, California, pp 149–160
29. Stutzbach D, Rejaie R, Sen S (2005) Characterizing unstructured overlay topologies in modern P2P file-sharing systems. In: Proceedings of the internet measurement conference. Berkeley, California
30. Tsoumakos D, Roussopoulos N (2003) A comparison of peer-to-peer search methods. In: WebDB, San Diego, 12–13 June 2003
31. Xiong L, Liu L (2003) A reputation-based trust model for peer-to-peer ecommerce communities. In: ACM conference on electronic commerce. San Diego, California, pp 228–229
32. Zhu Y, Yang X, Hu Y (2005) Making search efficient on Gnutella-Like P2P systems. In: IPDPS '05: proceedings of the 19th IEEE international parallel and distributed processing symposium (IPDPS'05) – papers. IEEE Computer Society, Washington, DC, p 56.1

Ioannis Pogkas received his BS in Computer Science in 2007 and is currently pursuing postgraduate studies at the Department of Informatics and Telecommunications of the Univesrity of Athens. His research interests focus on search, reputation and topology adaptation mechanisms in peer-to-peer networks. He is also interested in embedded and operating systems.

Vassil Kriakov received his B.S. and M.S. from Polytechnic University in 2001 and is now completing his doctoral studies at the Polytechnic Institute of New York University (NYU-Poly). His PhD research has been partially sponsored by a US Department of Education GAANN Graduate Fellowship. His research interests include distributed spatio-temporal data indexing, correlations in high-frequency data streams, and data management in grid and peer-to-peer networks.

Zhongqiang Chen is a senior software engineer at Yahoo! He holds a PhD in Computer Science and MS degrees in both Computer Science and Electrical Engineering all from Polytechnic University in Brooklyn, NY. He is a Computer Engineering MS and BS graduate of Tsinghua University, Beijing, P.R. China. He is interested in network security, information retrieval, and distributed computing and is the recipient of the 2004 Wilkes Award for outstanding paper contribution in The Computer Journal.

Alex Delis is a Professor of Computer Science at the University of Athens. He holds a PhD and an MS from the University of Maryland College Park as well as a Diploma in Computer Engineering from the University of Patras. His research interests are in distributed computing systems, networked information systems, databases and information security. He is a member of IEEE Computer Society, the ACM and the Technical Chamber of Greece.