

# Improving QoS on Gnutella

Daniel Hughes  
Computing Department  
Lancaster University  
Lancaster, UK.  
+44 (0) 1524 593793

d.r.hughes@lancaster.ac.uk

Ian Warren  
Computing Department  
Lancaster University  
Lancaster, UK.  
+44 (0) 1524 594117

iw@comp.lancs.ac.uk

Geoff Coulson  
Computing Department  
Lancaster University  
Lancaster, UK.  
+44 (0) 1524 593054

geoff@comp.lancs.ac.uk

## ABSTRACT

*The use of peer-to-peer systems for sharing information, files and other resources has risen dramatically over the last three years. File sharing is the 'killer app' that has driven this explosion in popularity. The first generation of peer-to-peer file sharing systems, including Napster, Morpheus and Kazaa, followed the traditional client-server paradigm. However, concerns over the legality and scalability of such systems has driven the development of entirely decentralized file sharing protocols, the most popular of these being Gnutella. To date, decentralized file sharing systems have been unable to compete with those based on a centralized architecture in terms of the QoS (quality of service) they offer to users. AGnuS improves QoS across the Gnutella network by increasing file availability, improving network friendliness, increasing file quality and improving file-acquisition time. These results are achieved by layering caching, load balancing, content-based routing and file filtering services on top of the core Gnutella protocol. AGnuS is implemented using the base Gnutella protocol in order to maintain compatibility with the large number of existing Gnutella hosts. Our experimental results show significant QoS improvements when compared to a network of generic Gnutella hosts.*

## 1. INTRODUCTION

Peer-to-peer communication formed the basis of the original Internet. IP routing is peer-to-peer as is Fidonet[1] and the original implementation of Usenet[2]. However, the explosion in home-users accessing the Internet via slow connections and low-performance PCs led to increasing division of functionality between client and server nodes. The computing power and bandwidth available to home users has increased significantly since the early days of the Internet and typical home computers now have resources to spare.

Peer-to-peer systems leverage unused resources around the edge of the network in order to provide services. In the case of Napster[3], users donate their bandwidth and disk space to provide a file sharing service. In the case of Seti@Home[4] and Folding@Home[5] users donate unused CPU time to a scientific experiment. Napster and

the clones which quickly followed its release (Kazaa[6] and Morpheus[7]) use a hybrid model of interaction. Servers are used as connection points and to keep track of users and files on the network. Since then, increasing scalability problems and legal issues have made the development of centralized services such as these increasingly unviable and the focus has shifted towards the development of fully decentralized peer-to-peer systems. We consider the semi-centralized file-sharing applications 1<sup>st</sup> generation and the fully decentralized file sharing applications 2<sup>nd</sup> generation. The most successful of the decentralized systems is Gnutella[8]. However, it suffers from a number of QoS issues due to the ad-hoc and self-organizing nature of the system.

### Scalability

The broadcast search mechanism employed by Gnutella does not scale effectively. Bandwidth consumption due to message passing increases dramatically as the number of users on the network grows[9]. For this reason Gnutella messages are assigned a time to live (TTL) value that limits how far they propagate through the network. However, this technique limits the number of nodes that queries reach. Typically, users have access to a 'search horizon' of around 10,000 nodes, which is less than 10% of the usual number of users present on the Gnutella network (Gnutellanet)[10].

### The Tragedy of the Commons

A rational Gnutella user with limited bandwidth and computational resources will always try to maximize the benefit they gain from using the network, donating little or no bandwidth to the community while downloading as much as possible. This behavior reduces the QoS experienced by other users across the network - a phenomenon referred to by Hardin as "Tragedy of the Commons"[11].

### Uneven Resource Distribution

Adar and Huberman[10] suggest that 70% of Gnutella nodes share no files at all, while 50% of files are served by just 1% of nodes. Consequently, the actual architecture of the Gnutellanet tends to a client-server model than a true peer-to-peer model. If one considers a typical search horizon of 10,000 nodes and factors in the user behavior

described by Hardin, one finds that users will typically have access to just 100 high-volume servers. This reduces fault tolerance as the failure of a small number of these nodes can have a significant impact on the QoS available across the network.

Furthermore, uneven resource distribution can lead to the problem of ‘flash crowds’[12] where dramatic and unpredictable rises in demand for a resource prevent some users from being able access a resource. Examples of this phenomenon involving the Internet include the posting of Ken Starr’s report in 1999, the Victoria’s Secret Web-Casts, and news coverage of the terrorist attacks against the United States on September 11<sup>th</sup> 2001. For a Gnutella network, flash crowds may cause files to be temporarily unavailable.

### Poor Quality files

Standard Gnutella nodes may freely distribute two types of poor quality files: masquerading files that simply waste bandwidth, and malicious files that threaten system security.

Masquerading files claim to contain one kind of data while actually containing another. These are a significant problem on peer-to-peer file sharing networks such as Gnutella. These files may be generated on-the fly by malicious nodes or they may be given a bogus file name at creation time. This behavior is used either to distribute unwanted content to nodes or to lower network QoS, and is engaged in by both individuals and organizations whose interests are contrary to those of the file-sharing community, e.g. ‘Media Defender’[13].

A significant number of Gnutella hosts have been observed generating spurious **QueryHit**<sup>1</sup> messages. These hosts respond to any **Query** they receive regardless of whether they are able to serve the requested file. When nodes attempt to initiate file transfer, a substitute file is served instead. This may be to ‘spam’ the network and reduce QoS or to distribute malicious files over a network.

Combinations of these QoS issues produce further emergent problems on the Gnutella network. For example a small number of high performance servers coupled with variable file availability give rise to ‘file-type skew’. The files served by hosts are generally specialized either towards a particular genre or a particular file format. Depending upon such specializations, one search horizon may generate more hits for a given **Query** than another.

Furthermore, the problem of flash crowds is particularly acute for nodes on low bandwidth links, such as dial-up connections. Unpredictable and substantial

increases in the number of **Query** messages being routed to a node on a low-speed connection can cause the node to be flooded with requests, resulting in lost messages in extreme cases. In addition, rapid bursts of incoming **Query** messages causes a node’s user to experience unpredictable response times for searches since the node may or may not be overwhelmed with incoming **Query** requests.

In this paper we describe and evaluate AgnuS’ QoS enhancing mechanisms. We also briefly discuss other relevant research and outline how QoS for decentralized peer-to-peer systems can be improved outside of the constraints imposed by the Gnutella protocol.

## 2. AGNUS

AGnuS was designed to address some of the shortcomings of conventional decentralized peer-to-peer systems as typified by Gnutella and introduced on Section 1. AGnuS is a specialized Gnutella host, which attempts to improve QoS across the entire Gnutella community, while simultaneously providing an improved service to its primary user. In short, AGnuS:

- *Increases the availability of files.* Using a caching scheme, supported by heuristic filters, and a content-informed routing policy, AGnuS locates a greater number of files in response to a **Query** request. This benefits the primary user since he/she gains access to more files of interest. In addition, the user community as a whole benefits because files are better distributed and navigated over the Gnutellanet.
- *Improves network friendliness.* AGnuS nodes employ a load balancing mechanism, which attempts to prevent a node’s available bandwidth being exhausted with **Query** requests. This benefits the primary user since his/her node retains bandwidth for generating its own search requests. Bandwidth is also conserved using heuristic filters that prevent unwanted files that might impose a security threat from being downloaded and distributed throughout a Gnutellanet. This offers benefit to the Gnutellanet community in addition to individual users.
- *Increases file quality.* Heuristic filters are also used to prevent users from being overloaded with files that are low quality and thus do not match what users are searching for.
- *Improves file acquisition time.* The collective set of mechanisms implemented in AGnuS leads to a reduction in the time taken to download files matching a search term.

AGnuS implements these improvements using the existing Gnutella protocol in order to maintain compatibility with the large number of existing Gnutella

---

<sup>1</sup> **Query** and **QueryHit** messages form part of the Gnutella protocol, which is described in [5].

nodes. AGnuS and standard Gnutella nodes can thus coexist in the same Gnutellanet. AGnuS employs four primary mechanisms to improve the QoS users experience across the network: load balancing, content based routing, caching and file filtering. Table 1 shows that these mechanisms are mutually supportive. For example, the content-based routing mechanism could impose an excessive amount of traffic on peers known to carry many files. Rather than penalise a node in this way for acting altruistically, the load balancing mechanism counteracts the potential for overloading peers with **Query** messages by ensuring a more even processing distribution. AGnuS is built on top of the Jtella base classes which allow the easy interfacing of Java applications to the Gnutella network[14]

	Increased file availability	Improved network friendliness	Increased file quality	Improved file acquisition time
Caching	✓			✓
Load balancing		✓		✓
Content based routing	✓			✓
File filtering		✓	✓	✓

**Table 1 QoS improvements and enabling mechanisms**

### Load balancing

AGnuS' load balancing mechanism contributes to both improving the speed of locating files and reducing bandwidth consumption. For the former, messages are routed away from peers that are responding to high volumes of **Query** messages and directed towards peers that are less heavily loaded. The mechanism thus favours peers that are more able to process search requests. In addition to improving file acquisition times, the load balancing mechanism attempts to ensure that a node's bandwidth is not completely exhausted by responding to incoming **Query** messages; since while a node is responding to **Query** messages, it has less bandwidth

available for initiating its own searches. Beyond these intended benefits, the mechanism also aims to reduce the variation in bandwidth consumption of a node, with the effect of providing more consistent and predictable response times to users.

The mechanism maintains connections with 8 peer nodes as opposed to 4, which is typical for standard Gnutella nodes. At runtime, the mechanism selectively routes search requests to 4 of the 8 connected peers. The choice as to which connections to route messages is made based on estimating the load of each connected peer. The 4 nodes that are deemed to be least loaded are selected.

The means of determining peer load is Jtella's built-in query monitoring systems. Using this function, the load of connected peers is measured in terms of the number of outgoing **Query** requests they generate. An AGnuS node periodically samples the rate of message generation to determine load. Ideally, would use a technique that takes into account the bandwidth capacity of a node's connection. In this way, it would be possible to discriminate between nodes on high-speed connections and those on low performance links, such as dial up connections. However, it is difficult to determine the connection speed of a peer while maintaining backwards compatibility with the Gnutella protocol which, as stated earlier, is a constraint of our work.

We arrived at 8 connections from which to choose 4 using experimentation. In general, there is a cost and a benefit which both increase as the number of connections increases. The cost of additional connections is an increase in bandwidth and CPU-time consumption, which is required to maintain each connection. The benefit of establishing additional connections is the number of potential search horizons increases, resulting in a greater number of peers to distribute load. The effectiveness of load balancing thus increases with the number of connections maintained. Based on trials, we have found that maintaining 8 connections offers an optimal trade off of bandwidth and effective load balancing.

### Content based routing

The content based routing mechanism works by directing search requests to nodes that are most likely to yield matching files. The consequent benefits of this mechanism include increased file availability and improved file acquisition time. Furthermore, the content based routing mechanism aims to combat the problems inherent in Gnutellanets, introduced earlier, of uneven resource distribution and file type skew.

By intelligently routing **Query** messages, an AGnuS node makes better use of available bandwidth since the number of hosts processing queries that they cannot fulfill is reduced. In addition, an individual node's processing becomes more efficient as it is more likely to respond to a

search request with a **Query Hit** itself or delegate the request to a peer which is more likely to find a matching file. The intended result is that more hits will be obtained while operating within Gnutella's constrained search horizon.

Similarly to the load balancing mechanism, this mechanism uses 8 connections and routes **Query** messages to the 4 connected peers that are deemed most likely to respond with matching files. The mechanism monitors the file density for a range of file types for each connected peer. This is carried out using wildcard searches, such as \*.mp3 and \*.avi for files of type audio and video respectively. An AGnuS node initiates a wildcard search periodically and records the number of corresponding **Query Hit** messages over a short period of time for each connection. The resulting file density value is then used in response to an incoming **Query** request to select the 4 connected peers with the higher density value for the type of file being searched. File filters, described shortly, are used to determine the type of file required by the incoming **Query** message.

### Caching

The caching mechanism aims to distribute popular files over a Gnutellanet with the result of increasing file availability and reducing the time required to locate files. In addition, the caching mechanism is intended to reduce the effects of flash crowds since the files are better distributed. Moreover, since files are more evenly distributed, AGnuS-based Gnutellanets tend to form peer-to-peer structures as opposed to client/server architectures that result from a combination of the problems outlined in Section 1. Consequently, the benefits of peer-to-peer systems are restored.

One of the key factors preventing users from sharing files on the Gnutellanet is that this consumes some of the finite resources available to them. With AGnuS, the caching mechanism enforces altruistic user behaviour by operating at times when the user is not using their computer. The mechanism continually monitors the popularity of files on the Gnutellanet and aggressively searches for, downloads and serves popular files, and thereby increasing the rate of file propagation across the network and reducing congestion.

File diffusion through the cache occurs passively as AGnuS nodes locate and download files from both AGnuS and Gnutella nodes. AGnuS nodes continually monitor current searches on the Gnutellanet and from these searches, a list of the most sought-after content is compiled. From this compilation, a ranked list of potential downloads is created, where the most popular files are assigned the highest priority. AGnuS nodes search for and download these files in priority order. However, downloading all files that match a popular search term is

not an efficient caching method. To maximize bandwidth and caching efficiency, heuristic filters are employed to select the most promising download candidates.

### Heuristic filtering

Heuristic filtering improves file quality by removing malicious and masquerading files from further consideration by AGnuS nodes. The caching mechanism does not cache files that are identified as poor quality. Users can also be warned of poor quality files based on analyzing data obtained from filters. In addition, bandwidth consumption is reduced since once a file is tagged as a poor quality file, bandwidth is not wasted on downloading it. AGnuS nodes perform a set of heuristic rules on files such as comparing a **Query** request's file type with the likely size of a file of that type and format.

Beyond addressing the problem of poor quality files, filtering is used to support the content-based routing mechanism by identifying the type of file to be searched from a search query. The filtering mechanism works at 3 levels:

- *Primary.* At this level, incoming **Query** messages that contain file-type information are scanned and appropriately categorized.
- *Secondary.* Heuristic analysis of **Query** requests is be used to identify their file type based on the presence of signature sub-strings. For example, television episodes frequently contain series and episode information in plain text format. s01e01, 1x01 or 101 could all be used to represent episode one of the first season of a television show. Similar signatures exist for many other media types and are used to determine the type of file required.
- *Tertiary.* Where **Query** requests do not contain file-type information or signature strings, **Query** messages are processed by comparing them with previously categorized searches. If a significant similarity exists between the search-term being analyzed and previously identified search terms, queries are assumed to be of the same type.

## 3. EVALUATION

In this section we evaluate AGnuS in terms of the QoS improvements introduced in Section 2: increased file availability, improved network friendliness, increased file quality and improved file acquisition time.

### Increased file availability

To evaluate the extent to which the combination of content-based routing and file caching increase file availability, we conducted a test using an AGnuS node and a standard Gnutella node. The two node types were connected to a

Gnutellanet for a 24 hour period. At 2 hour intervals, each node initiated a Query search and the number of corresponding QueryHit messages was recorded. This period and sampling rate were chosen to gather measurements over a representative time of Gnutella usage. For both node types, the search term was the same: “DivX AVI”. The rationale for using this search term was that it was the most popular **query**, excluding searches for pornographic and copyright infringing material, at the time the test was conducted.

	AGnuS node	Standard node
00:00	64	3
02:00	53	31
04:00	190	235
06:00	49	38
08:00	115	7
10:00	244	103
12:00	27	24
14:00	116	58
16:00	118	90
18:00	125	89
20:00	126	3
22:00	140	20
24:00	170	226

**Table 2 Results for increased file availability**

Table 2 shows the test results and reveals that throughout the test period the AGnuS node consistently located more files than its standard Gnutella counterpart. The average number of hits received per search was 76 for the standard Gnutella node and 128 and for AGnuS node. This equates to a 68% improvement in file availability when using AGnuS over a standard node.

The results thus support our hypothesis that the combination of content-based routing and caching improves file availability. For the Gnutella community, this means that files are more effectively distributed throughout the network and problems associated with flash crowds and indispensable servers are reduced. For individual users, selective routing compensates for file-type skew and exploits multiple search horizons.

**Increased network friendliness**

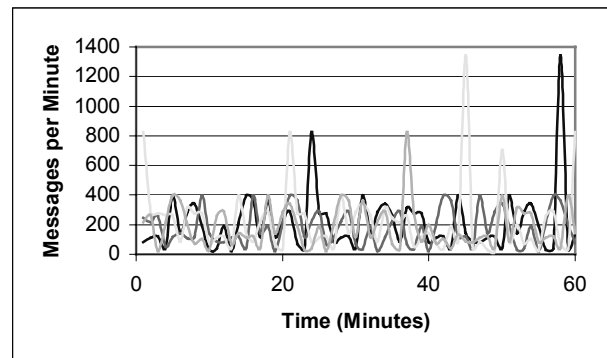
Earlier, we introduced the notion of network friendliness and claimed that AGnuS is network friendly with respect to bandwidth consumption and security. For the latter, malicious files are detected and not distributed throughout the Gnutellanet. This is beneficial to both the primary user and the Gnutella community as a whole. Bandwidth

consumption is reduced in AGnuS since it detects poor quality files and these are not downloaded. We defer evaluation of improvements in file quality until a little later; here we focus on the extent to which AGnuS attempts to regulate a node’s bandwidth consumption with the aim of providing a more predictable response time to users running queries and preventing nodes from being flooded with **Query** messages.

We conducted three one-hour tests, once using an AGnuS node and then repeating the three tests using a standard Gnutella node. Over the course of each hour, samples of host load were taken at the rate of one a minute. For each trial, the node under test was connected at a different place within the Gnutellanet.

Figures 1 to 3 show the variation in load for peers of a standard Gnutella node. Each connection is shown on a separate connection. Similarly, Figures 4 to 6 show the load variation for peers of an AGnuS node. These graphs show that using a standard Gnutella node, load varies significantly. In contrast, the load on an AGnuS node’s peers is much less volatile. Quantitatively, the average standard deviation in host load for Figures 1 to 3 is 223, 177, and 182 respectively. Using an AGnuS node, the corresponding average standard deviations are much lower: 89, 92 and 93. Overall, this gives an average reduction in host load variation of 40% when using an AGnuS node.

The effects of AGnuS’ load balancing mechanism are twofold. First, users of an AGnuS node experience a more predictable system since their node’s bandwidth is not consumed by unpredictable bursts of incoming **Query** requests. Thus, when conducting searches, response times are likely to be more consistent. Second, AGnuS nodes are less likely to be flooded with incoming message that are lost. While flooding cannot be guaranteed, the leveling effect imposed by AGnuS reduces the risk of flooding. In the absence of information about a peer’s bandwidth capacity, flooding cannot be prevented.



**Figure 1 Load variation of a standard node’s peer (1)**

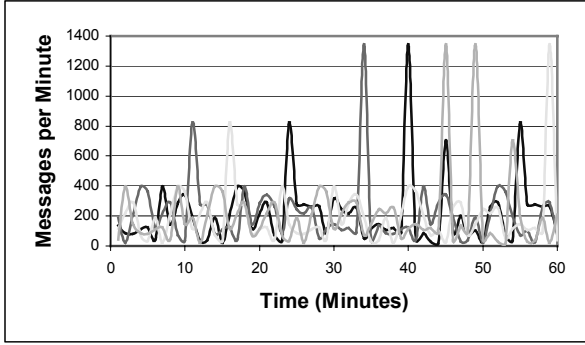


Figure 2 Load variation of a standard node's peer (2)

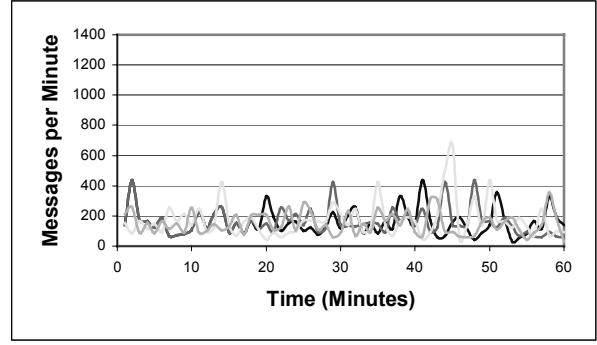


Figure 6 Load variation of an AGnuS node's peer (3)

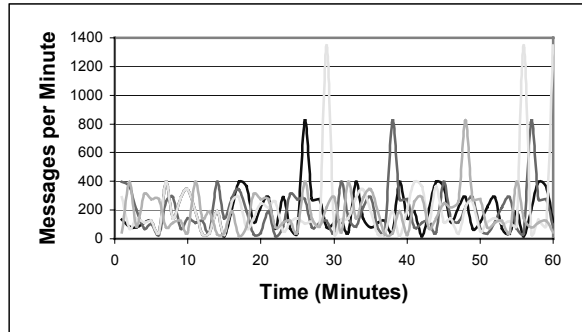


Figure 3 Load variation of a standard node's peer (3)

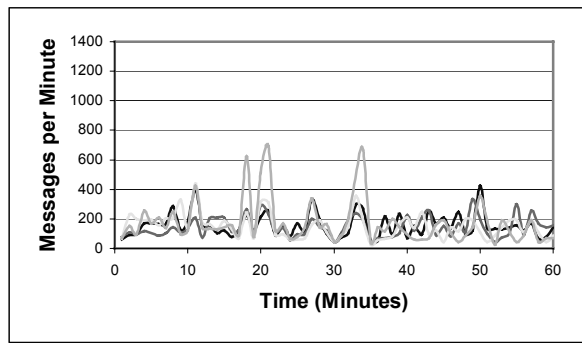


Figure 4 Load variation of an AGnuS node's peer (1)

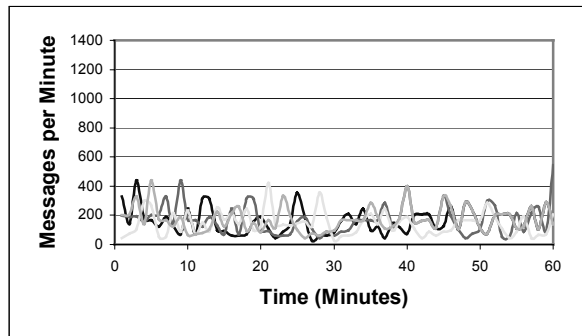


Figure 5 Load variation of an AGnuS node's peer (2)

### Improving File Quality

To measure the effectiveness of AGnuS' improvement in the quality of files downloaded, we tested an AGnuS node with popular search terms spanning multiple file types. We ran the tests three times with the node connected to a different point in the network for each test. We then repeated the tests using a standard Gnutella node.

We used popular search terms since it is reasonable to expect that these would be picked for effectively distributing masquerading and malicious files. We chose several file types to reveal the filtering mechanism's behaviour with respect to multiple file types. In particular, we used the following search terms:

- *Audio* Eminem.
- *Video* Star Wars.
- *Software* Microsoft.
- *Text* Harry Potter.

Table 3 shows the results of running the tests. The figures are percentages that show the proportion of files downloaded that match the search term. In all cases, AGnuS' filtering mechanism significantly increases the proportion of quality files that are downloaded. The increase in quality for each file type is: 22% for audio, 82% for video, 16% for software and 70% for text.

The variable effectiveness of the heuristics is due to the variable initial quantities of quality files. For example: A much higher percentage of audio files are genuine than video files.

Furthermore, some file types are naturally easier to analyze than others. For example: Size is used as a condition for filtering video files of different types but the likely size of a piece of software is much difficult to ascertain.

Standard Gnutella node				
	Audio	Video	Software	Text
T1	67%	5%	10%	40%
T2	75%	7%	12%	20%
T3	56%	9%	10%	33%
AGnuS node				
T1	88%	67%	25%	100%
T2	92%	100%	30%	100%
T3	85%	100%	25%	100%

**Table 3 Results for improved file quality**

### Improved file acquisition time

We conclude the evaluation with some comments on how file acquisition time is improved using AGnuS. The improvements gained are the result of AGnuS' cooperating QoS mechanisms.

The caching system increases the speed of file propagation across the network, and thus reduces the problems associated with congestion and flash crowds. As more instances of a file are available, there is less competition for access to the same file instance in a Gnutellanet. Consequently, the length of time at which a download request is queued at any one host that offers the file is likely to be reduced. Thus, from caching alone, the time taken to acquire a file is likely to be improved.

Table 2 shows that content-based routing contributes to improved file availability. Given that a **Query** request made using an AGnuS node will locate the required file on more nodes than if a standard node were used, a user has more choice in terms of node to download the file from. There is thus a higher probability of the user finding a node on a relatively fast connection from which to initiate the download.

Finally, filtering improves file acquisition times by preventing time being wasted by the download of malicious or masquerading files. The download of such files wastes bandwidth, and therefore time, for both the Gnutellanet and for the individual user. The effectiveness of the filtering mechanism is summarized in Table 3.

## 4. RELATED WORK

In addition to Gnutella, other relevant peer-to-peer systems include Pastry [15], Freenet [16] and Ultrapeers [17].

Similarly to Gnutella, Pastry is based on an entirely decentralized structure. Pastry adapts well to the arrival, failure and departure of nodes and uses a scalar proximity metric to minimize the distance that messages travel. Significantly, when presented with a key and a message,

Pastry efficiently routes the message to the node whose unique ID most closely matches the key. Although efficient, this routing mechanism is unsuitable for general file sharing systems where the destination of a **query** request is unknown.

Freenet is a decentralized distributed file storage system which attempts to ensure replication of files, user privacy and information integrity. Freenet's primary focus is information security. However, Freenet does not counter the scalability issues that apply to networks of this type. Furthermore, such a system could not be implemented on the Gnutellanet while maintaining backwards-compatibility with existing Gnutella hosts which is a constraint of our work.

Ultrapeers are designed to increase Gnutella scalability by consolidating Gnutella clients under Ultrapeer hosts. The inherent scalability limit to the network structure of Gnutella nodes still applies, but as each Ultrapeer serves a cluster of leaf nodes, more peers can be reached. While clustering of this form increases scalability, it introduces single points of failure into the network (the Ultrapeers). In addition, the numerous compatibility issues between the different schemes used have so far prevented smooth integration across the Gnutella network.

## 5. CONCLUSIONS & FUTURE WORK

In this paper, we have reported on AGnuS, a Gnutella node which remains backwards compatible with the Gnutella protocol. AGnuS improves the QoS experienced by both individuals and the Gnutellanet community as a whole; it is thus likely to be acceptable to rational users.

AGnuS offers the following QoS improvements using four mutually supportive mechanisms:

- *Increased file availability.* For the individual user, AGnuS' content-based routing mechanism significantly increases file availability by selectively routing queries to peers that are most likely to satisfy the queries. Users receive more hits for a given **query** using AGnuS, as opposed to a standard Gnutella node. For the Gnutellanet community, AGnuS' caching mechanism increases the speed of file propagation across the Gnutellanet, thus reducing congestions problems.
- *Improved network friendliness.* AGnuS nodes employ heuristic filters which lead to a reduction in bandwidth consumption by precluding poor quality files from being downloaded, and which result in both the community and individuals being protected from (malicious) files that can compromise security. In addition, AGnuS' load balancing mechanism regulates **query** traffic to peers. This reduces the probability

that peers will be flooded with incoming requests and offers more predictable response times for individuals.

- *Increased file quality.* Heuristic filters address the problem of information overload where users are presented much information, some of which is irrelevant. AGnuS uses filters to automatically filter out low quality files and prevents these from being cached and downloaded.
- *Improved file acquisition time.* As a consequence of the above QoS improvements, users can acquire files more quickly using AGnuS nodes than standard nodes. In short, improved resource distribution, more effective use of bandwidth, and greater choice of nodes from which to download files from contribute in a reduction in the time required to acquire files.

Gnutellanets can potentially be used for applications other than file sharing. One example is distributed searching as in Infrasearch [18]. However, while such systems work in the lab, user behavior on the Gnutellanet causes problems for many such applications. Indeed, this is one of the reasons Infrasearch switched to the Jxta[19] middleware platform. The mechanisms implemented in AGnuS counteract the poor QoS offered on Gnutella, so that it could support such applications.

The level of QoS that Gnutella can offer is nevertheless restricted by the underlying Gnutella protocol. We therefore advocate the design of an extended protocol, which uses a more generic and comprehensive model to provide support for QoS assurances in addition to static and dynamic adaptation. Fundamental to our plans is a departure from the purists' peer-to-peer model in which all nodes are equal (for example Pastry and Freenet). In practice, individual nodes differ greatly in terms of their resources and consequently in terms of the QoS they can offer. For example, a typical peer-to-peer system may comprise PCs with low bandwidth connections in addition to high performance servers connected to the network using high-speed links. Such a model requires the following capabilities:

- *Resource Awareness.* Using reflection, a node's peers can discover its available resources. As shown in our experiments with AGnuS nodes, awareness of the resources available on hosts can be used to improve QoS experienced across the network; AGnuS achieves QoS improvements by routing according to the resources available on general areas of the network. Routing on a host-by-host basis has the potential to produce a far greater improvement.
- *Capture and Discovery of Operational Profiles.* To complement static resource information, operational profiles may be used to capture run-time information, such as likely periods of connectivity. This information

contributes to QoS assessment and allows for more effective QoS reasoning.

- *QoS Reasoning.* Based on acquired resource information, a node can assess the volume and quality of responses that it is likely to receive from its peers and make informed choices about which peers should be selected to provide a service. A node can change its own behavior based on resource information and QoS judgments, allowing it to contribute more effectively to the network as a whole.

## 6 REFERENCES

- [01] **Fidonet** <http://www.ftsc.org/>
- [02] **Usenet** <http://www.karlsruhe.org/rfc/rfc977.txt>
- [03] **Napster.** [www.napster.com](http://www.napster.com).
- [04] **SETI@Home.** [setiathome.ssl.berkeley.edu](http://setiathome.ssl.berkeley.edu).
- [05] **Folding@Home.** [www.foldingathome.com](http://www.foldingathome.com)
- [06] **KaZaA.** <http://www.kazaa.com>
- [07] **Morpheus** <http://www.musiccity.com>
- [08] **The Gnutella Protocol Specification,** <http://dss.clip2.com/GnutellaProtocol04.pdf>
- [09] J. Ritter, **Why Gnutella Cant Scale – No Really** <http://www.letto.net/docs/gnutella-cant-scale.html>
- [10] E. Adar and B. A. Huberman, **Free Riding on Gnutella.** *Fist Monday Oct. 2000.*
- [11] G. Hardin, **The Tragedy of the Commons.** *Science volume 162, pp. 1243-1248, 1968.*
- [12] L. Zhang, S. Floyd, and V. Jacobson, **Adaptive Web Caching,** *Proceedings of the 1997 NLANR Web Caching Workshop, April 1997*
- [13] **Media Defender.** [www.mediadefender.com](http://www.mediadefender.com)
- [14] **Jtella** <http://jtella.sourceforge.net/>
- [15] A. Rowstron, P. Druschel **Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,** *Lecture Notes in Computer Science, 2001.*
- [16] I. Clarke, O. Sandberg, B. Wiley, T. Hong, **Freenet: A Distributed Anonymous Information Storage and Retrieval System,** *Lecture Notes in Computer Science 2000.*
- [17] A. Singla, C. Rohrs. **Ultrapeers – Another step towards Gnutella Scalability** <http://rfc-gnutella.sourceforge.net/Proposals/Ultrapeer/Ultrapeers.htm>
- [18] **Infrasearch,** <http://www.gonesilent.com>
- [19] Li Gong, **Project JXTA: A technology overview** Technical report, Sun Microsystems, October 2001.