

Improving QoS for Peer-to-Peer Applications through Adaptation

Daniel Hughes, Ian Warren and Geoff Coulson

Computing Department, Lancaster University,

Lancaster, LA1 4YR. UK

d.r.hughes@lancaster.ac.uk | iw@comp.lancs.ac.uk | geoff@comp.lancs.ac.uk

Abstract

The use of peer-to-peer networks has increased dramatically in recent years. As the power of home computers and Internet access speeds increase, we envisage that an increasingly diverse set of applications will seek to make use of the growing pool of resources available around the edge of the network. We suggest that existing peer-to-peer networks are unable to cater for the diverse demands of future applications. We discuss how adaptation may be used to improve the QoS of existing peer-to-peer networks and introduce an innovative network structure with inherent support for multiple levels of adaptation.

1. Introduction

With the increasingly ubiquitous provision of broadband access such as ADSL and the ever-increasing power of home PCs, there exists an increasing pool of resources available around the edge of the network. As this pool of resources expands, it is likely that more diverse applications will seek to take advantage of these untapped resources, using peer-to-peer networking. We envisage that these may include ad-hoc distributed computation, streaming media on demand and multimedia group working. Each class of application raises different QoS issues. We believe that in heterogeneous peer-to-peer environments, where nodes have very different capabilities and requirements, adaptation is essential to support the QoS requirements of diverse applications.

In general, to accommodate the diversity in emerging peer-to-peer applications, and to address a range of consequent QoS issues, an underlying model for hosting peer-to-peer applications should offer:

- The ability to reliably reach any node connected to the network.
- The ability to discover new resources on the network.
- Adaptive behavior to compensate for the highly variable nature of peer-to-peer nodes.
- Extensibility for supporting emergent application requirements.

In section 2 we discuss our experiences with implementing adaptive networking enhancements on the Gnutella [1] network with the Altruistic GNUtella Server (AGnuS) [2]. In section 2.5 we describe the QoS improvements achieved by layering adaptive networking behaviour on top of the core Gnutella

protocol. This work is constrained to be backwards compatible with the Gnutella protocol and to operate using Gnutella's underlying unstructured decentralized network. In section 3, we continue by reviewing other existing architectural models for peer-to-peer systems and critique them in terms of the requirements discussed earlier for managing diversity. In section 4, we introduce a model which provides inherent support for multiple levels of adaptation, and which aims to better address the diverse needs of future peer-to-peer applications. Finally, in section 5 we conclude and identify avenues of further work.

2. **AGnuS: Improving QoS on Gnutella through awareness and adaptation**

The Gnutella network (Gnutellanet) is an unstructured decentralised overlay network [4] based upon a Cayley tree [3]. Resources are discovered using a broadcast search mechanism. Incoming search messages for one peer are forwarded to the node's connected peers. These peers behave similarly and forward search requests to their connected peers, the ultimate effect being that queries are broadcast through the network. Any node which receives a search request may respond with a message which is routed back to the node that initiated the query. Gnutella suffers from the following QoS issues:

Scalability: The broadcast search mechanism causes bandwidth usage to rise dramatically with the number of queries. To counter this, queries are assigned a time to live (TTL) value that limits how far they propagate through the network. With the default Gnutella TTL value of 7, users will typically have access to 10,000 nodes – typically a small fraction of the number of nodes available on the network. This set of nodes is referred to as the 'Search Horizon' for a given node.

Harmful user behaviour: Rational users with limited resources will always try to maximize the benefit they accrue from using the network, donating few of their limited resources. This behavior reduces QoS for each user of the network; a phenomena known as the "Tragedy of the Commons" [5].

Uneven Resource Distribution: Adar and Huberman [6] suggest that 50% of all files on Gnutella are served by just 1% of nodes, making the actual architecture of the Gnutellanet closer to a client-server model than a true peer-to-peer system. This introduces single points of failure and can cause problems due to flash crowds [12] which result in resource unavailability.

Poor Quality files: Gnutella may distribute two varieties of poor quality files: masquerading files that simply waste bandwidth, and malicious files that also threaten system security.

AGnuS seeks to improve the Quality of Service (QoS) users experience on Gnutella using adaptive networking. Specifically, AGnuS seeks to:

- Increase file availability across the network.
- Increase network friendliness.
- Improve file quality.
- Reduce file-acquisition time.

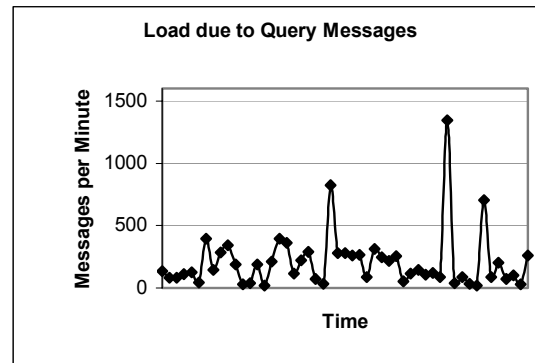
AGnuS employs four mutually supporting mechanisms to provide the above QoS improvements. Table 1 summarises which mechanisms address each QoS improvement. A design constraint on this work was that AGnuS should remain backwards compatible with the core Gnutella protocol, in order for Gnutella nodes to interoperate with AGnuS nodes.

	Increased file availability	Improved network friendliness	Increased file quality	Improved file acquisition time
Caching	✓			✓
Load Balancing		✓		✓
Content Based Routing	✓			✓
File Filtering		✓	✓	✓

[Table 1 – AGnuS QoS enhancing mechanisms]

2.1 Load Balancing

Host load due to message passing varies greatly on the Gnutellanet. Figure 1 shows the rate at which a Gnutella host is requested to route query messages. The changing amount of bandwidth consumed by query messages can significantly affect the experience of users on slow connections such as GSM or GPRS. Load balancing in AGnuS is implemented by connecting a node to a higher number of peers than usual:



[Figure 1 Query Load Processing]

eight connections are maintained instead of the original four used in Gnutella. However, messages are routed to only four hosts at any one time. An AGnuS node scans connected peers using Gnutella's SearchMonitorSession construct and adapts its routing behaviour based on the load of connected nodes. If hosts are found to be under relatively low load, a greater proportion of messages will be routed to them. Conversely, if hosts are found to be under high load, fewer messages will be routed to them. The load balancing mechanism thus aims to prevent nodes on low-capacity connections from being flooded with messages they have no capacity to process.

2.2 Content Based Routing

Content based routing enhances the QoS of the Gnutellanet in two ways: first, the number of hosts processing queries that they cannot fulfill is reduced. This makes better use of available bandwidth and processing efficiency across the network. Second, as queries are forwarded to those hosts that are most likely to be able to fulfill them, more hits can be obtained within the constraints of the Gnutellanet search horizon. Each AGnuS node connects to its eight peers and periodically monitors the density of different file-types available through each connection where type density is the relative abundance of each format within that peer's search horizon. Based on file-type density, the node adapts its message forwarding behaviour for incoming requests based on their type. File density awareness within each

search horizon is measured using wild-card searches (e.g. *.mp3, *.avi and *.mpg). The number of QueryHit¹ messages returned within a known period following the generation of such queries is interpreted as being proportional to file density. Incoming queries are categorised by the file filtering system as described in section 2.4.

2.3 Caching

AGnuS is designed to run at times when users would not otherwise be using their computer. AGnuS users benefit from automated downloads, while the Gnutellanet benefits from file caching.

AGnuS nodes continually monitor current searches on the Gnutellanet in order to build awareness of the most sought-after content. The system then adapts its caching behaviour to encompass popular downloads, where the priority of a download is proportional to its popularity. This ensures that popular resources are rapidly disseminated across the Gnutellanet, thereby avoiding the problems of flash crowds.

2.4 File Filtering

The filtering system allows for the categorisation of incoming queries by type and for the screening of download candidates. This is important as a significant portion of the files on Gnutella are either malicious or masquerading. Blind caching of such files would only aid their propagation across the network. File filtering is accomplished using a filtration heuristic with three discreet levels:

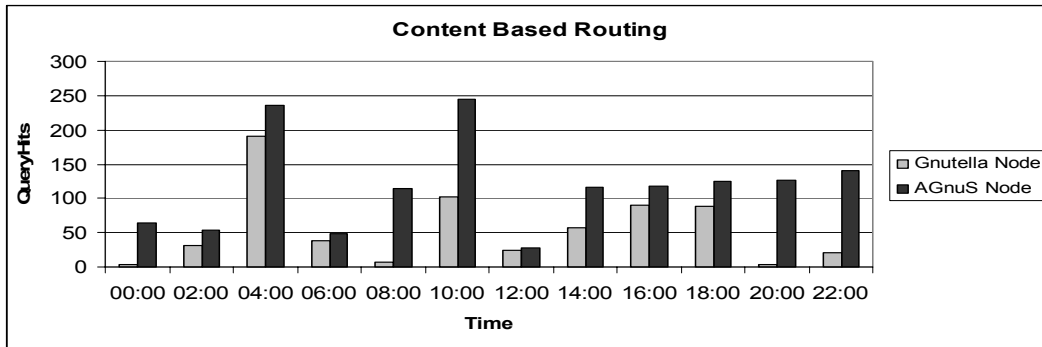
- **Primary** categorises incoming Queries based on included file type information.
- **Secondary** categorises files based on the presence of signature sub-strings.
- **Tertiary** categorises queries by comparison with previously categorized searches.

2.5 Evaluation

In this section we evaluate the extent to which AGnuS is able to meet its objectives.

Increased file availability To evaluate the extent to which content-based routing increases file availability, we conducted tests using both AGnuS nodes and standard Gnutella nodes. The two node types were connected to a Gnutellanet for a 24 hour period. At 2 hour intervals, each node initiated a search for the most popular current file on Gnutella and the number of corresponding QueryHit messages was recorded:

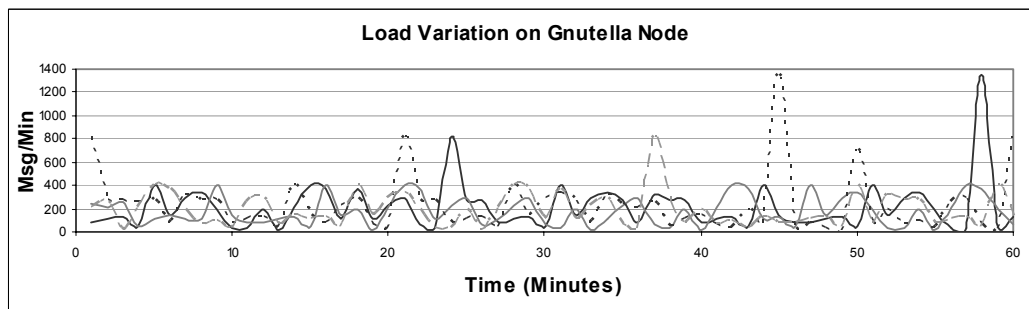
¹ A QueryHit message is sent in response to a search query and forms part of the Gnutella protocol.



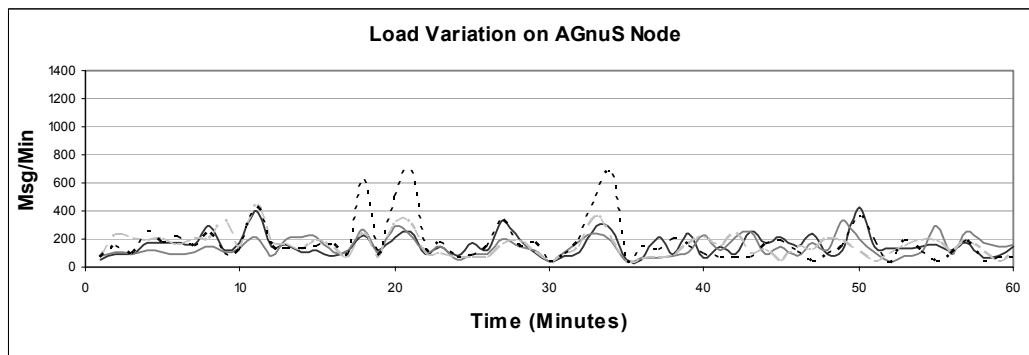
[Figure 2 – Increase in QueryHit responses due to Content Based Routing]

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 the AGnuS node. AGnuS nodes provide 68% improvement in file availability over a Gnutella node through the use of content based routing. The mutually supportive nature of AGnuS’ mechanisms means that nodes with popular content are not penalised by the content based routing system as it directs more requests to them; as load climbs to an abnormal level, the load balancing system simply routes messages away from the node.

Increased network friendliness: Here we focus on the effectiveness of AGnuS for regulating a node’s bandwidth consumption with the aim of providing a more predictable response time for handling queries and preventing nodes from being flooded. Three one-hour tests were conducted using an AGnuS node and then repeated using a standard Gnutella node. The graphs below show the changing load on immediately connected AGnuS and Gnutella peers:



[Figure 3 – Unbalanced load variation]



[Figure 4 – Balanced load variation]

The load on an AGnuS node's peers is much less volatile, with the average standard deviation in host load being reduced from 194 queries per minute to just 91. AGnuS nodes offer a 56% reduction in load variation when compared to a standard Gnutella node, which reduces the possibility of nodes on low-bandwidth connections being flooded with queries.

Improving File Quality: To measure the effectiveness of AGnuS' file filtering system, 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.

In all cases, AGnuS' filtering mechanism significantly increases the proportion of quality files that are downloaded. File filtering increases the number of quality files downloaded by: 22% for audio files, 82% for video files, 16% for program files and 70% for text files. The high degree of variation in improvement is due to a combination of the efficiency of the heuristics available and the relative availability of file types on the network.

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 2 – Quality files downloaded (%)]

3. Discussion

The mechanisms implemented in AGnuS go some way towards counteracting the relatively poor QoS offered on Gnutella. However, the level of QoS that Gnutella can offer is inherently restricted by Gnutella's underlying unstructured decentralised network. This model facilitates resource discovery, subject to the search horizon constraint introduced earlier. In addition to this model, there are two other established models that are used to host peer-to-peer applications: semi-centralised and structured decentralised networks.

Semi centralized models as typified by Napster [7], SETI [8] and Kazaa [9] are based on central servers. In the case of Napster, such servers are indexing servers which store information about network nodes and the resources they are sharing. This approach is scalable, so long as there is a central authority which is capable of maintaining the necessary servers. Using this model, nodes play the client role in requesting an introduction to other nodes. From this point on, nodes are peers that are able to interact directly with one another. With semi-centralized networks, central servers are single points of failure; the failure of a single server can severely reduce the number of resources that can be discovered. Furthermore, the use of a centralized authority prevents ad-hoc resource sharing communities from being formed.

Structured and decentralized models such as Pastry [10] and Chord [11] implement a distributed hash table which provides for efficient and scalable message routing. Each network resource is assigned a unique key. Using its key, a particular resource can be contacted anywhere on the network. However, the key of a resource must be known beforehand, which makes resource discovery difficult. Attempts to build search functions onto Pastry have so far focused on the development of distributed indexing servers [13]. While this provides resource discovery, it reduces the peer-to-peer nature of the network and introduces the same legal concerns as encountered by Napster.

Table 3 shows that existing peer-to-peer models meet only some of the requirements presented in section 1. Furthermore, existing systems' routing algorithms tend to be fixed: node positioning within a network is arbitrary and routing behaviour is static. We think that there is potential for using adaptive routing algorithms and networks that are structured into regions based on node properties as part of a solution for dealing with diverse application demands on heterogeneous peer-to-peer networks.

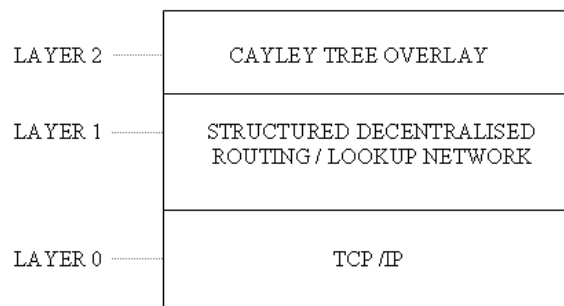
	Structured Decentralised Networks	Unstructured Decentralised Networks	Semi- Centralised Networks
Reachability of all nodes	✓		✓
Resource Discovery		✓	✓
Adaptive Behaviour			
Support for Ad-Hoc communities	✓	✓	

[Table 3 – Comparison of Architectural Models]

4. Hybrid system model

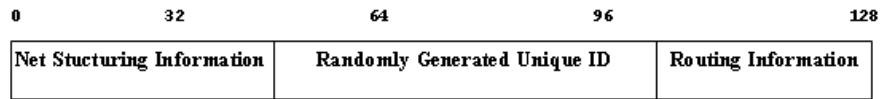
We envisage that it will be advantageous to provide support for multiple levels of adaptation: network restructuring, routing behavior adaptation and peer-selection adaptation. We propose a hybrid model which integrates the unstructured and structured decentralized models to enable the first two kinds of adaptation. Peer-selection adaptation is concerned with nodes discriminating between other nodes when determining which node is most suited to interact with. We view this latter form of adaptation as being something which can subsequently be added to the hybrid model we describe here. Figure 5 shows that the model comprises three layers. At layer 0, TCP/IP is used as the underlying transport.

More interestingly, layer 1 uses a distributed hash table to provide a structured decentralized network. The purpose of this is to enable efficient message routing and to guarantee reachability of all nodes. This network is overlaid, in layer 2, with an unstructured decentralized network which adds a broadcast search facility.



[Figure 5: Network stack]

The basic abstraction for routing messages (layer 1) is a variant of the Pastry routing algorithm. Our algorithm differs from Pastry's in that we use *reflective* keys. Keys are reflective since they embed meta-data about the nodes they represent. This is in contrast to Pastry where the 128 bit keys are simply pseudo random numbers. The rationale for using reflective keys is to support networks that can be structured into regions of similar nodes and to support adaptive routing algorithms.



[Figure 6: Reflective key structure]

Figure 6 shows the structure of a reflective key. The most significant bits of the key are used for structuring a network into regions. These are the bits that contain meta-data about the node to which the key is assigned. The intermediate bit sequence is used to ensure uniqueness of keys, much in the way that all bits in a basic Pastry key are used. The remaining, least significant bits embed additional meta-data that has little impact on where a node is positioned but are used by adaptive routing algorithms. This model is intended to support a diverse variety of applications. We now describe the benefits of the model for a particular class of application: resource sharing.

4.1 Example: resource sharing

In Section 2, we described the notion of a search horizon which is the result of trading resource discovery for scalability in unstructured decentralized networks. With our model, a search horizon remains but we hypothesize that the nodes that fall within the search horizon are the “right” set of nodes to target with a search request.

For example, with a file sharing application, the meta-data encoded in a key's most significant bits can identify a particular node as one that shares files in addition to the type of files it specializes in. Once such a key has been generated, the underlying Pastry algorithm will join the node to the network in a region that also contains other nodes that specialize in sharing similar files.

Subsequently, when a search request for a particular type of file is submitted, the system will use the information contained in the request to generate a key. Since the most significant bits of the key are used to embed this information, the key will target a node that is in the most appropriate region of the network to successfully deal with the request. At this point, the layer 2 network is used to broadcast the search request to other nodes within the same region.

Earlier we reported on Adar and Huberman's study which revealed that 70% of nodes share no files at all, while 50% of all files are shared by 1% of nodes. With a search horizon of 10,000 nodes (i.e. 10,000 reachable nodes), these studies suggest that only 3,000 of these nodes will actually be sharing files of any kind. Using our system model, we can expect a three-fold increase in locating nodes that share files because the search horizon is populated with file sharing nodes. Furthermore, we expect the

actual increase to be greater still because the nodes targeted will not just be sharing files, but will be sharing the type of files targeted by a search request.

5. Conclusions

In this paper we have reported on the AGnuS project which has significantly improved QoS for Gnutella by layering adaptive network behaviour on top of the core Gnutella protocol. AGnuS offers the following QoS improvements using four mutually supportive mechanisms:

- Increased file availability.
- Increased Network Friendliness.
- Increased file quality.
- Improved file acquisition time.

However, we feel that Gnutella's underlying architectural model prevents further QoS improvements. We are now developing a more generic and comprehensive model for enhancing QoS on peer-to-peer systems through enhanced support for resource awareness and adaptation.

To this end, we have begun work on a hybrid system, based on a combination of two proven models, whose aim is to host a diverse set of peer-to-peer applications. Essentially a structured decentralized model provides efficient routing on which a broadcast search mechanism is layered. We have shown that the model uses meta-data encoded in keys to support network restructuring where similar nodes can be located in a particular area of the network. We have hypothesised that the model can offer improvements over existing systems in terms of hosting traditional peer-to-peer applications such as resource sharing.

Our hybrid model can also be used to adapt routing behaviour. For example, consider a peer-to-peer chat application where all message passing is handled by peer nodes. As peers must route all network messages, participating in this kind of community on mobile low bandwidth connections, such as GPRS and GSM, would be expensive and would use an unreasonable portion of a node's available bandwidth. Routing adaptation can be informed, similarly to network structuring, by meta data embedded in the reflective keys. However, in the case of adaptive routing, the least significant bits of the key should be used to embed routing data so as not to significantly affect the positioning of the node in the network. From interpreting routing meta data, nodes can cease routing requests to peers on low bandwidth connections. The effect of such adaptive routing would be to allow nodes on low bandwidth connections to use services cost effectively and without being swamped with routing requests. Furthermore, other nodes would benefit since their messages would be routed by peers connected using faster and more reliable connections.

In addition to experimenting with adaptive routing, our further work is also concerned with accommodating a node that participates in the network in different ways. For example, a node might share video files and also participate in a chat service. Using regions to the structure the network, it seems that the node might be placed in two separate regions. We are investigating the use of multiple

overlay networks where a node on the same physical connection can appear in multiple regions simultaneously.

6. References

- [01] The Gnutella Protocol Specification: <http://dss.clip2.com/GnutellaProtocol04.pdf>
- [02] Hughes D, Warren I, Coulson G. "AGnuS: The Altruistic Gnutella Server", proceedings of the Third international conference on peer-to-peer Computing, Linköping Sweden, 2003.
- [03] Rains E, Sloane N; Cayley's Enumeration of Alkanes. *Journal of Integer Sequences*, 1999.
- [04] Walkerdine J, Melville L, Sommerville I. Dependability Properties of P2P Architectures, proceedings of the Second international conference on peer-to-peer Computing, Linköping Sweden, 2002.
- [05] Hardin G. The Tragedy of the Commons. *Science volume 162*, pp. 1243-1248, 1968.
- [06] Adar E, Huberman B. Free Riding on Gnutella. *Fist Monday Oct. 2000*.
- [07] Napster. www.napster.com.
- [08] SETI@Home. setiathome.ssl.berkeley.edu.
- [09] Kazaa. www.kazaa.com
- [10] Rowstron A, Druschel P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, *Lecture Notes in Computer Science*, 2001.
- [11] Stoica I, Morris R, Karger D, Kaashoek M, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for Internet applications. Technical Report TR-819, MIT, March 2001
- [12] Zhang L, Floyd S, Jacobson V, Adaptive Web Caching, *Proceedings of the 1997 NLANR Web Caching Workshop*, April 1997
- [13] Reynolds P, Vahdat A. Efficient Peer-to-Peer Keyword Searching, *Middleware 2003*, Rio de Janeiro, Brazil. June 2003