# Connecting P2P to the Web: A Gnutella-WWW Gateway*

Brian D. Davison, Wei Zhang and Baoning Wu
Department of Computer Science & Engineering, Lehigh University
19 Memorial Dr. West, Bethlehem, PA 18015 USA
{davison,wez5,baw4}@cse.lehigh.edu

**Abstract**

We present an approach to extend the abilities of peer-to-peer networks to take advantage of broad information resources on the World Wide Web and vice versa. Either network may contain the data to satisfy a searcher's information need. Our work improves the accessibility of files across different delivery platforms, making it possible to use a single search modality. We propose a gateway between the WWW and the Gnutella peer-to-peer network that permits searchers on one side to be able to search and retrieve files on the other side of the gateway. The design and implementation of such a gateway is described, along with access statistics from test deployments and lessons learned.

## 1 Introduction

The search for information can take many forms. The availability of multiple kinds of search tools both helps and hurts this process — it can narrow the search to a smaller collection (when knowledge of the collection contents is available), but it may also require the searcher to perform multiple steps, moving from one tool to the next in the quest to satisfy some information need. For example, in years past, one might have used archie [4] to search FTP archives. While a web search engine might also index files accessible by FTP, it is rare. Thus, a searcher might need to use different tools to find the file or files of interest.

Today it is common to provide access to files by putting them on a Web server, or by sharing them within a peer-to-peer network. Some files may only be available on one platform or the other, and so searchers are thus required to use multiple tools to be able to do an exhaustive search.

Our work is designed to improve the accessibility of files across different delivery platforms, making it possible to use a single search modality. We propose

*Technical Report LU-CSE-05-016, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 18015.

a gateway [16] between the WWW and the Gnutella peer-to-peer network [6] that permits searchers on one side to be able to search and access files on the other side of the gateway.

In this work we present the design and implementation of a novel gateway system to extend the searching abilities of users of the Gnutella network to the Web domain, and vice versa. We describe why an organization would want to operate such a gateway, and have put our prototype implementation to use in scenarios that reflect some of those motivations. We report our conclusions from those tests, and argue that this system can help address a user's information need.

The rest of this paper is organized as follows. We first motivate the use and deployment of P2P-WWW gateways in Section 2, and provide background for a Gnutella gateway in particular, as well as related work in Section 3. We describe the system architecture and implementation in Section 4. In Section 5, we present statistics and discuss lessons drawn from test deployments of our system, including the number of queries, responses and downloads. Finally, we summarize our contributions in Section 6.

# 2 Motivation

Operating a P2P-Web gateway may consume substantial resources as query traffic can be considerable, and often file downloads must additionally flow through the gateway. This raises the question as to whether one might want to run such a gateway, other than for research purposes. We have identified multiple motivations to deploy such a gateway to today's peer-to-peer networks.

- **Personal gateway.** An individual may wish to operate a generic P2P-WWW gateway for personal use (or to share with friends). Some kind of access control may be needed to prevent (or regulate) general usage, but an otherwise lightly used gateway will be operable within many users' resource constraints.
- **Organizational gateway.** Instead of an individual, an organization might operate a gateway to improve the accessibility of information to its members. Access control is again necessary, but when the resources used are all to the benefit of an organization's members, it can be justified.
- **Intranet gateway.** Instead of connecting to a generic WWW search engine, an organization might connect the gateway to an intranet search engine so that the resources searched are all internal.
- **Propaganda distribution gateway.** This gateway, like the above, might search a limited collection (e.g., corporate product literature), but be world-accessible to promote the distribution of that literature. Such gateways could be operated by any information provider who benefits (perhaps indirectly) from the distribution of particular information. Possible examples would be hardware vendors distributing datasheets on their

products, or programming language vendors distributing language documentation.

- **Paid search gateway.** Companies that offer paid search results (such as Yahoo's Overture[1] and Google[2]) would likely welcome a new market for their services as such vendors are paid when links from their results are retrieved. Typically such companies will encode the result link so that they can track individual click-throughs.

- **Advertising-based search gateway.** Companies that offer advertiser-supported search results (all the major search engines) could benefit from operating a gateway that returned links to interstitial advertisements, rather than directly to the requested content. However, a feasibility study may be required to determine user acceptance of such a system.

- **Legitimate use gateway.** As a result of the current legal climate, RIAA[3] opponents might choose to operate such a gateway to introduce additional legitimate (non-copyright-infringing) content to peer-to-peer searchers to defend such networks' existence.

The above scenarios predominantly describe when to operate a P2P-WWW gateway; in most cases, the use case for a WWW-P2P gateway is analogous. For example, while a company might not wish to provide global access to P2P search facilities through the Web, it may wish to do so for internal use. We will describe experiments (below, in Section 5) that implement two of these gateway scenarios.

# 3 Overview

## 3.1 Peer-to-peer networking

Peer-to-peer file sharing provides an important channel to allow users to share their own files and retrieve information from remote machines. Since most information comes for free in P2P systems, more and more users turn to P2P systems to search documents, music, movies, etc.

Gnutella is one of the first P2P networks, and as an open protocol, has been studied by many researchers (e.g., [1, 12, 10]). While it once grew by a factor of 25 in a six-month interval in late 2000 and early 2001 [10], more advanced P2P networks have since been developed and participation in Gnutella has largely stabilized, with a network size of around 200,000 over most of 2003 [8] but has since continued to grow.

Peer-to-peer systems are decentralized, and nodes will freely join and leave the network on a frequent basis. Thus, a central concern is how to locate available content. In Gnutella, queries are distributed using a naive message

[1] http://www.overture.com/
[2] http://www.google.com/
[3] The Recording Industry Association of America, http://www.riaa.com/

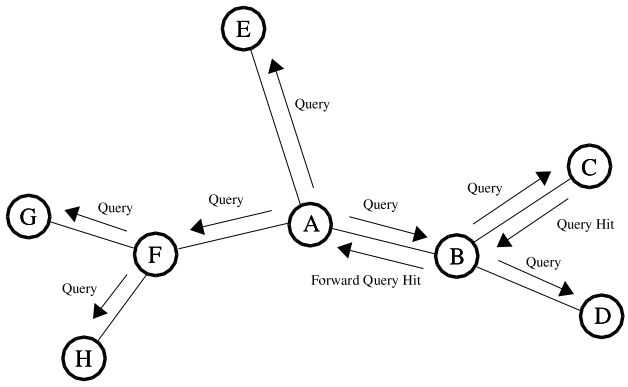


Figure 1: Search in Gnutella Networks. Node A sends a query to its neighbors (B, E, F), who re-broadcast the query to their neighbors. Node C has a matching object for node A's query, and so returns a query hit message to node B, who forwards the result back to A.

flooding approach. More efficient protocols (e.g., in Pastry [11] and Chord [15]) to find specific files have since been proposed and variants adopted in newer P2P networks.[4] When a query arrives at a Gnutella node, it searches locally and passes the query to its neighbors (where the process is repeated until the query has traveled a maximum number of hops). As shown in Figure 1, if files match the query, a query hit message will be generated and passed back; then, it continues flooding the query to its neighbors, and TTL (time-to-live in terms of the number of hops remaining) and count of hops traveled by the query message are updated accordingly. While P2P networks are popular, their cooperative basis opens them to the dilemma of free riding [1] in which users exploit the network resources without making similar (or any) contributions back to the network.

### 3.2 A P2P network gateway

As we suggested in the Introduction, it is sometimes difficult to satisfy a user's information need with any single collection of information. The WWW is huge, and yet its servers may not contain the same kind of content that individual users share on a P2P network. Likewise, queries on P2P networks may be satisfied with content from the Web.

A gateway (also sometimes called a protocol converter) enables communication between networks that use different protocols. This is exactly the purpose of our system—to convert between the WWW and Gnutella, within the context

[4]However, these protocols typically need to match a filename (or identifier) exactly, thus limiting the ability to search.

of search and retrieval. Our gateway operates both as a Gnutella client and server (a.k.a. a servent) and as a Web server and client. It captures Gnutella queries that it receives as a member of the network, and forwards them to a search engine. The gateway takes the results from the search engine responses and forms Gnutella messages to transfer them back to the searcher via the P2P network. Similarly, when the gateway receives a query through its Web interface, it distributes the query to all of its Gnutella neighbors. As results are collected from the network, they are presented to the Web searcher.

In addition to searching files, the gateway also helps searchers retrieve the files from the alternate network. Although Gnutella uses HTTP for file transfer, additional constraints on the URL specification make a Gnutella client unable to retrieve an arbitrary URL from the Web. In the other direction, a Web browser is able to generate a correct retrieval request for a Gnutella servent, but most servents will refuse such requests from browsers to discourage users that are not participating in the network. Thus, it is necessary to relay data from the Web to the Gnutella network (and vice versa) to accommodate acceptable request formats. By providing bi-directional search and retrieval services, the gateway provides significant benefits for both Gnutella users and Web surfers, and also addresses P2P networkers' concerns regarding free riding [1] (since each of the networks connected via the gateway will now benefit from the content available on the other).

Since a P2P-WWW gateway lives in two networks, it must respect both. Thus, a good gateway design will consider many aspects, including coverage (serving as many users as possible), responsiveness (to respond promptly to queries), and efficiency (both internally, and externally by not abusing available resources). In Section 2, we provided a variety of motivations for deployment, and so we expect that gateways such as ours can be deployed at many boundary points between the WWW and P2P networks like Gnutella.

### 3.3 Related Work

While our system is unique in terms of its functional completeness, the general idea of a Gnutella-Web gateway is not novel. Here we describe prior and related work.

The LimeWire Peer Server[5] provided LimeWire Gnutella clients with the ability to query additional data sources, such as e-commerce databases. In comparison, our system has several different features: our gateway has bidirectional data flow, and users can take advantage of our gateway to download files (in whatever form they exist).

Most prior work provided features for the other direction. In past years, there have been Gnutella search services that provided a Web interface for users to enter a query, send that query through Gnutella, and return results via the Web. Links on the result page would go directly to the Gnutella servents, allowing the searcher to retrieve files directly. This type of Web search no longer works —

[5] http://www.limewire.com/english/content/peerserver.shtml

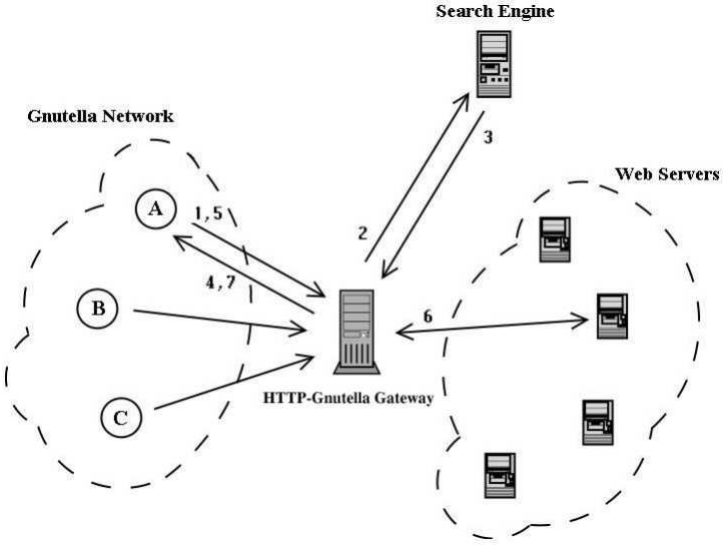


Figure 2: Operation of the Gnutella-WWW gateway.

most Gnutella servents will reject browser clients with a message asking users to contribute to the network by running Gnutella software.

For example, the DISCOVIR system [14] helps users find image files within the Gnutella peer-to-peer network. It includes a Web interface to permit WWW users to issue queries. While our system can also find P2P-hosted images (albeit not as easily) as well as any other file type, our system also provides the reverse — allowing P2P users to find Web-hosted content. Additionally, our system will assist in the process of retrieval, as well as search.

FileDonkey[6] and Jigle[7] are search engines that can help users locate files in the eDonkey P2P network. FileDonkey employs a network crawler to collect the information of files available on the eDonkey network. Jigle, in contrast, performs real-time network searches and caches the results for up to three hours in case of additional requests using the same query. However, neither of them help Web users download files, but instead require that users use one of several eDonkey client packages to download, such as eDonkey2000[8] and eMule[9].

# 4 Implementation

We have implemented a P2P-WWW gateway with a prototype that supports Gnutella and the Web. We modified an existing open-source Gnutella servent,

[6] http://www.filedonkey.com/
[7] Available from 2002 to early 2004 at http://jigle.com/
[8] http://www.edonkey2000.com/
[9] http://www.emule-project.net/

Gtk-Gnutella[10], version 0.91.1, to implement our gateway system. Gtk-Gnutella supports both versions 0.4 and 0.6 [7] of the Gnutella protocol. Instead of searching a local filesystem when a query is received, our system searches the Web.

## 4.1 Details of our approach

### 4.1.1 Gnutella-to-WWW

The operation of our gateway is illustrated in Figure 2. Gnutella node A broadcasts a query message (1), which is received, translated, and forwarded by the gateway to a Web search engine (2), which generates a set of query results and returns them (3). The gateway translates the results into Gnutella formats and then forwards them to node A (4), such that results are available through the gateway. If node A sends a download request to the gateway (5), the gateway will fetch the data from the Web server (6) and then forward the data to node A (7).

Figure 3 shows the system architecture for search and retrieval of WWW data via Gnutella. The gateway resides on both the Gnutella network and the Web. The gateway core is the coordinator, which is responsible for translating and passing messages to particular modules. The Gnutella download module is used to accept downloading requests from the Gnutella clients, and send back file data from the Web site. The web download module sends HTTP requests to the Web servers and receives data from them. These data will be transferred to the Gnutella download module and then to the Gnutella clients. The Web search engine interface is employed to help search Web links on a specific Web

[10]http://gtk-gnutella.sourceforge.net/

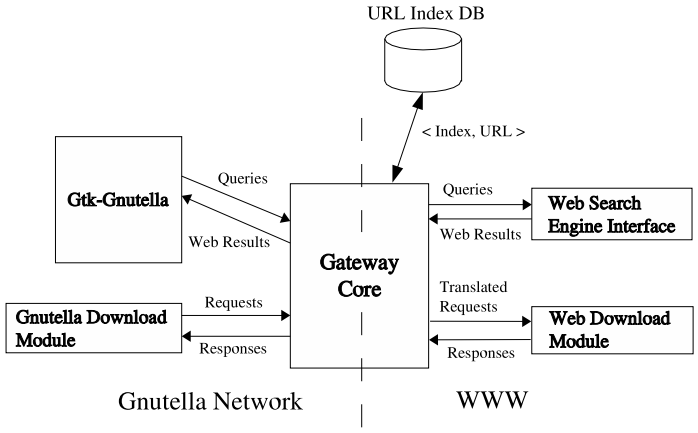


Figure 3: Gnutella-WWW Gateway Architecture

| Time | MUID | IP address | T | Query |
|---|---|---|---|---|
| 02831 | 1541447684-141... | 212.40.X.X | Q | tomb raider |
| 31201 | 2611702049-263... | 68.103.X.X | Q | marketing tips rtf |
| 30576 | 1056034299-114... | 66.75.X.X | Q | funny videos |
| 30680 | 2596152080-380... | 66.75.X.X | R | http://www.akjedsvcs.com/ELEMCAT2002.pdf |
| 36515 | 577604398-286... | 12.223.X.X | R | http://www.usd308.com/staff/cat1.pdf |

(a) Google

| Time | MUID | IP address | index | T | Query |
|---|---|---|---|---|---|
| 91707 | 2367762773-105... | 162.33.X.X | 238524 | R | http://www.homeschoolbookdepot.com/ |
| 91708 | 2367762773-105... | 162.33.X.X | 238538 | R | http://www.reggaemovement.com /Artists/prezidentbrown.htm |
| 91753 | 0-2332295168-416... | 66.32.X.X | 0 | Q | die another day |
| 92490 | 730901841-184... | 217.133.X.X | 0 | Q | charlies angels 2 |

(b) Overture

Table 1: Search log samples.

search engine. It forms an HTTP request from a Gnutella query and sends it to the search engine, and then parses the returned HTML page to extract links.

A downloading request from a Gnutella client requires an index number as well as the file name (an example is shown in Section 4.2.3). Since we don't directly provide Gnutella clients with a URL, we instead compose a file name from the title of the Web page. In order to serve a file retrieval request, we record the URL along with a unique index value into a local database. Thus, when a request arrives the gateway, we retrieve the URL from the database via the index provided by the Gnutella client.

Since we employ Web search engines for back-end query processing, recklessly forwarding many P2P queries may affect their service and even lead to a perceived Denial-of-Service attack.[11] Therefore, we enforce a minimum delay between query transmissions (placing delayed queries into a queue).

The Gnutella network can be thought of as a huge mesh consisting of a great number of nodes, in which a node only connects with its logical neighbors. Thus, increasing the number of links supported by the gateway can augment the coverage. On the other hand, too many links lead to high CPU usage and slow response time. Our gateway currently supports approximately two dozen simultaneous connections.

Our prototype generates two separate log files: one for Web search and response activity; the other is for recording download request and completion events. For the first, the log file records timestamp, message identifiers, neighbors' IP addresses, URLs, queries, etc. For the second one, it records timestamp, IP addresses of the download initializers, file size, URLs, HTTP response codes,

[11] In fact, our first implementation (described in Section 5) managed to send enough queries to Google that our system's IP address was blocked for months afterward.

status of completion, etc. Examples of the logs from these experiments are shown in Tables 1 and 2. The redirection in the downloading log stands for how many times the link is redirected to the destination page. The status code can be 0, 1 and 2: 0 means the attempt to download fails; 1 means it's a redirection; 2 means it was successfully downloaded.

To validate our gateway, we first verified that a few Gnutella clients (Gtk-Gnutella and LimeWire[12]) were able to connect, send queries, and download response links from the gateway. We additionally made certain that the retrieved files were intact, by using `diff` on the P2P-retrieved files and an additional copy retrieved directly from the origin server using the `WGet` utility [9].

#### 4.1.2 WWW-to-Gnutella

In addition to serving Gnutella clients, our gateway also permits Web users to query and retrieve resources in the Gnutella network. In this direction, a Web user submits a query via a Web interface, which is transmitted to the Gnutella servents connected to the gateway. As results are received from the Gnutella nodes, the gateway compiles the results and presents a hit list to the Web searcher, specifying the gateway as the source of those files. If the Web searcher attempts to download a file, the gateway will extract the original servent address and filename and contact that node and initiate the download, passing the contents back to the Web searcher as they are received.

Since the gateway maintains a large number of links to its neighbor nodes, queries can be quickly flooded to a large set of nodes. Gnutella does not have a maximum bound on query response times, so in order to generate a Web page within a reasonable amount of time, we arbitrarily collect and present responses received within six seconds. In the future we plan to make the trade-off explicit to the user by allowing the searcher to select among a range of search intensities (which would provide for smaller or larger query TTLs and corresponding response-time thresholds).

### 4.2 Issues

A number of issues were raised during the design and implementation of our Gnutella-WWW gateway. Here we describe those issues and present our solutions.

#### 4.2.1 Web search engine interface

Some search engines, like Google, provide a search engine API that can help exploit their searching abilities. However, we wanted a generic interface that can be adapted to any engine, and so we utilize the default search engine front-end interface by sending queries as Web requests, and parse the links from the generated results page. In addition, under normal conditions the number

[12] http://www.limewire.com/

of requests to search engines are typically quite limited.[13] Obviously, when a gateway is deployed by the company operating or developing the back-end engine, query limits are no longer an issue, and more efficient (proprietary) communication interfaces can be put into place.

#### 4.2.2 Query filtering

Queries from Gnutella are a bit different from those from Web users. Many queries contain a file type as query term, such as *mp3*, *avi*, etc. As Web search engines mainly provide links to HTML resources, we ignore queries with those file types. Moreover, too short or too long queries are also ignored because search engines will likely return ambiguous results or otherwise no results for these queries. By filtering out improper queries, the gateway system can more efficiently and fairly serve nodes that are likely to be interested in Web resources.

#### 4.2.3 Downloading

Besides searching in the Web, the gateway also helps Gnutella clients download files from Web servers. As mentioned earlier, the Gnutella network uses HTTP for file retrieval, but the URL is partially constrained. A typical Gnutella download request header is in the form of:

```
GET /get/1234/foo.mp3 HTTP/1.1
User-Agent: Gnutella
Host: 123.123.123.123:6346
```

Since most Web servers would interpret this request as specifying the file to be within the subdirectory /get/1234/ where 1234 is the file identifier, this kind of request would not successfully retrieve the intended file. So the gateway translates it slightly as:

```
GET /foo.mp3 HTTP/1.1
User-Agent: Gnutella-WWW Gateway
Host: www.foo.com
```

#### 4.2.4 URL-index database

As shown above, the node must know the index in order to issue a download request. So the gateway assigns an index number to each URL response. The index number along with URL, priority and query is stored in a small database. When the gateway gets a download request, it searches the mappings to find

[13] For the Google API, the number is limited to 1,000 per day, corresponding to approximately 1.4 requests per minute.

| Time | IP address | URL | Size | Code | Success |
|---|---|---|---|---|---|
| 089224 | 200.67.X.X | http://www.martinfowler.com /articles/coupling.pdf | 2625 | 200 | 1 |
| 116384 | 80.131.X.X | http://www.biochemj.org /bj/366/0265/3660265.pdf | 230819 | 200 | 1 |
| 140321 | 24.160.X.X | http://www.artsalive.ca /en/infozone/pdf/Handel_all_e.pdf | 4262 | 404 | 0 |

(a) Google

| Time | IP address | Index | URL | Redir | Success |
|---|---|---|---|---|---|
| 60697 | 208.186.X.X | 57463 | http://www.marketing-register.com /pages/Marlon_Sanders1.shtml | 1 | 2 |
| 59955 | 63.250.X.X | 54228 | http://launch.yahoo.com /artist/news.asp?artistID=1043454 | 1 | 2 |

(b) Overture

Table 2: Download log samples.

the URL by consulting the index of the request, and then goes to download the file.

#### 4.2.5 File size

While Web browsers don't need the file size before downloading a file, the Gnutella client must know it when a query hit occurs. However, the gateway is unable to accurately know the file size when Web links are returned from search engine, and incorrect file size will lead to downloading failure. To compensate, we implemented two solutions for different cases.

In one scenario, only HTML pages are requested. We set a fake size for each query hit message. If the real file size is less than the fake size, spaces will be padded in order to fill the fake length; if the real file size is greater than the fake size, the file data will be forwarded to the Gnutella client until the fake size of bytes have been sent and then the rest will be trimmed. In the second case, Gnutella client will not see an intact page, but carefully setting a fake size can guarantee most pages are wholly passed through without too many extra padding spaces. Therefore, we arbitrarily set the fake size to 52572. In Section 5.3.2 we examine other values for file sizes.

In a second scenario, all requested files are of type PDF, so the exact file size is needed as padding will be unsuccessful. Before sending a query hit message to Gnutella client, the gateway initiates a HEAD request to the Web server storing the PDF file. If the response contains the entry of "Content-Length" in the response header, then the size is obtained and sent to Gnutella clients along with the query hit message; otherwise, the response message is discarded.

#### 4.2.6 Additional handshaking

The gateway for downloading uses a different port from the Gtk-Gnutella base. When the gateway is running, there are two ports bound to the gateway. Some Gnutella nodes will think that they download files from a new node, so they will use the Gnutella handshake messages before sending the downloading request, while some other nodes may think they have already connected to the gateway, so they send the downloading request directly to our gateway. Thus when it receives a message from a node, the gateway needs to check if it is a *Hello* message or a direct downloading request, and then handle them accordingly. If a *Hello* handshake message is received, the gateway has to send back an *OK* message, then the node knows it has been successfully connected and sends the downloading request shortly.

#### 4.2.7 Access control

A number of the deployment motivations provided earlier include restricting access to the gateway. For Web servers, access control can be through the use of authentication or permitted IP ranges or domain names. In Gnutella, however, queries are passed anonymously — the recipient does not know who sent them. As a result, access control in Gnutella is more difficult, but some mechanism is still needed.

One could limit access to particular IP address ranges for immediate neighbors, but to ensure that only the (vetted) immediate neighbor's queries are processed, one would also need to reject queries with hop counts greater than 1. In contrast, the approach we used to help control demand for our system was to prioritize access. For example, we assign a priority for each query packet from immediate neighbors. The priority can be based on particular IP range so that the clients in the range will obtain responses more promptly. Moreover, other conditions, such as the number of hops traveled, and/or TTL remaining, can also be used to prioritize access.

## 5 Experiments

We operated test gateways running under Linux to perform our experiments. The startup process of the gateway is the same as normal Gnutella servents — it always tries to connect to the IP addresses in the local cache. The gateway system ran at different periods and the accumulated runtime is roughly one month. We ran one using Google in May of 2003, and another using Overture from October to November of 2003.

### 5.1 Scenarios

We employed two scenarios to investigate users' interest in the system.

- The first scenario was to build a generic gateway between Gnutella and the WWW, using Google as the back-end search engine. This is similar

Table 3: Gateway Usage Statistics

| **Personal Gateway with Google** | |
|---|---|
| No. of queries | 319,245 |
| No. of served queries | 113,391 |
| No. of responses | 930,860 |
| Average query response time (seconds) | 16.33 |
| No. of download requests | 952 |
| No. of different IP addresses | 67 |
| No. of served download requests | 945 |
| No. of successfully served download requests | 740 |
| Total size transfered (bytes) | 244,227,881 |
| Average response time (seconds) | 3.15 |
| Average total download time (seconds) | 15.92 |
| **Paid Search Gateway with Overture** | |
| No. of received queries | 806,876 |
| No. of served queries | 37,011 |
| No. of query responses | 861,103 |
| No. of successful downloads | 324 |

to a personal gateway, except that we let all the world connect to us. We believed that Gnutella searchers are typically interested in files (and not in Web sites), so we used Google's advanced feature to specify PDF as the required file type.

- The second experiment was specifically intended to implement a paid search gateway. For this we selected Overture as the Web search engine, because it only serves paid results, and represents an interesting potential business model to motivate real deployment of our system. Results from Overture were links to web pages.

While these are only a few of the potential scenarios described in Section 2, they provided representative situations which required slightly different implementations. Elsewhere [3], we report on an additional deployment of our gateway in an intranet scenario. While we implemented a public WWW interface to search and retrieve files from the Gnutella network, we did not publicly deploy it since we could not expect automatic use of it (unlike our Gnutella-side interface to the Web).

## 5.2 Results

Each of these scenarios generated usage logs, which we analyze and from which we report statistics below.

### 5.2.1 Personal gateway scenario

The first experiment was conducted by using the Google search engine. After running the gateway for almost 5 days, we get the usage statistics shown in

the upper portion of Table 3. We can see that during these 5 days the gateway received in total 319,245 query messages of which it processed a little more than a third. This difference is a result of our query filtering and queuing mechanisms. Since it may send several query hit messages for one query message, the gateway sent out a total of 930,860 query hit messages. We also calculate the average response time, that is, the average time difference between when the gateway system received the query message and when it sent the first query hit message for the query message. The average time is 16.33 seconds, which includes the queuing delay in the gateway, the time of searching on Google and forming the query hit message, and the time spent getting the file size via sending HEAD requests to the Web servers. We also recorded how many users sent downloading requests to our system and how many files each of them retrieved through our system, and how long it takes to finish the downloading operation.

From Table 3, we can tell that during our experiment, we got 952 downloading requests from 67 different IP addresses. The gateway tried to satisfy 945 requests, but only succeeded in 740 requests. Here success means that we got the file from the specified web server and passed it to the requested Gnutella node successfully, and we only count the HTTP response code 200 or 206 as successful; we consider 404 as unsuccessful. Nearly 244MB of data passed through our system to Gnutella nodes. The average response time is the mean time difference between when we received the downloading request and when we begin to send the first packet of the file to the node. Because we do not cache files in our system and only pass through packet contents, it takes some time to send the whole file because it must be retrieved from the origin server. So the average total download time is almost 16 seconds which is much bigger than the average response time 3.15 seconds.

#### 5.2.2 Paid search scenario

The other experiments employed the Overture search engine. In this case, the gateway only provides the Gnutella nodes with HTML page search and download. The results are shown in the lower portion of Table 3.

The number of served requests are much lower than that of received requests because we apply more strict criteria for query filtering. Many unsuitable queries are filtered out. It can improve the quality of the responses. For example, some users want to download mp3 files, but Overture only provides HTML pages instead of other file types. Thus the returned pages for such users may not be useful.

### 5.3 Analysis

#### 5.3.1 Active clients of Google system

We selected the top twenty clients with the largest number of files downloaded from our system. The distribution is shown in Figure 4. Each bar represents a unique IP address and the vertical axis represents the number of files downloaded

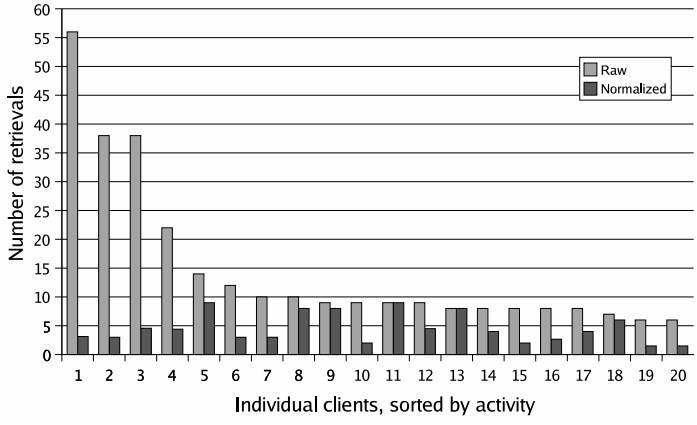


Figure 4: A comparison of the raw number of downloads for the most active clients in the Google trace, versus the mean number of downloads per unique query (the normalized case).

from this IP address. The figure is drawn in the descending order. The first machine downloaded 56 files, and several other machines also downloaded close to 40 files. Such large activity from a few users suggested that perhaps these were automated systems, downloading all responses provided (perhaps looking for copyrighted material). To explore this further, we additionally plot in Figure 4 a normalized number of downloads — the number of retrievals per query. This we see that client 11 retrieved the files for all nine responses, client 13 did likewise for all 8 responses, and clients 8, 9, and 18 retrieved almost all responses provided. But the really active clients only retrieved perhaps a third of the responses provided. Thus, we believe it unlikely that much of our logged activity is from automated systems. In general, Figure 4 demonstrates that the results provided through our system are attractive to at least some Gnutella users.

### 5.3.2 Distribution of downloaded file sizes

In Section 4.2.5 we mentioned the arbitrary selection of 52572 bytes for results. The value was chosen to be both large enough to span the likely file sizes of returned HTML pages, but also with the expectation that P2P users were likely to be interested in larger files (representing better or more detailed content).

We tested this hypothesis in an additional experiment by randomly varying the size of result files. We assigned a file size uniformly at random from values between 10KB and 210KB. This version of our system was put into place for approximately 3 days, receiving more than 700,000 queries, from which our system selected more than 32,000 from which to generate responses. The distribution of sizes for the 69 file retrievals is displayed in Figure 5 (after placing files into

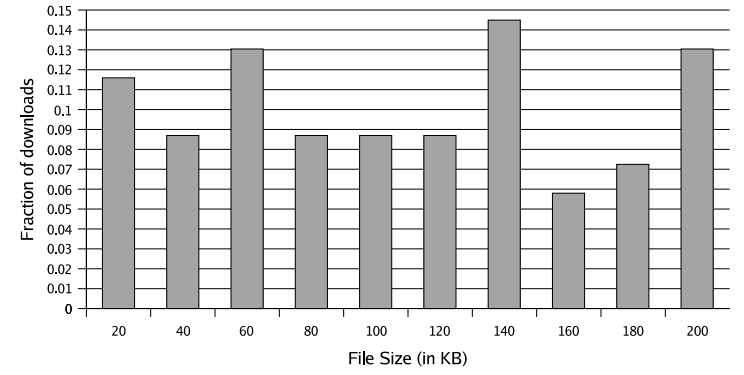


Figure 5: Distribution of the user-retrieved file sizes.

20KB buckets). The figure shows a relatively even distribution, suggesting that our hypothesis was false, and that users do not place significant emphasis on the size of the file (for files $\leq$ 210KB) when choosing what to retrieve (at least not for our relatively unique files).

### 5.3.3 Response time vs. retrievals

Our prototype implementation queues P2P requests that cannot be immediately satisfied. As a result, the overall response time of a query can vary. We hypothesized that P2P searchers would be more likely to retrieve a file when the response time of the query was short. We explored this hypothesis by analyzing the relationship between the response time and the number of downloads performed for results having a particular response time (that is, the time between when we received the request and when we sent out the first response). Figure 6 plots the distribution of response times, and a normalized value of the popularity of results at each response time. These figures demonstrate that users clearly prefer to download files from responses that arrive quickly. In fact, if the response time is larger than 12 seconds, almost no retrievals are performed. From this, we can easily determine an appropriate threshold age after which the system should remove a query from the queue.

## 6 Summary

We have presented the design and implementation of a novel gateway system to extend the searching abilities of Gnutella participants to the Web domain, and vice versa. Users from one network are also able to obtain files from nodes on the other network via the gateway. We have tested our prototype under multiple scenarios to demonstrate the potential of our approach. Furthermore, we have enumerated a number of potential motivations for commercial deployment, including distribution of information and pay-per-retrieval models. Under these

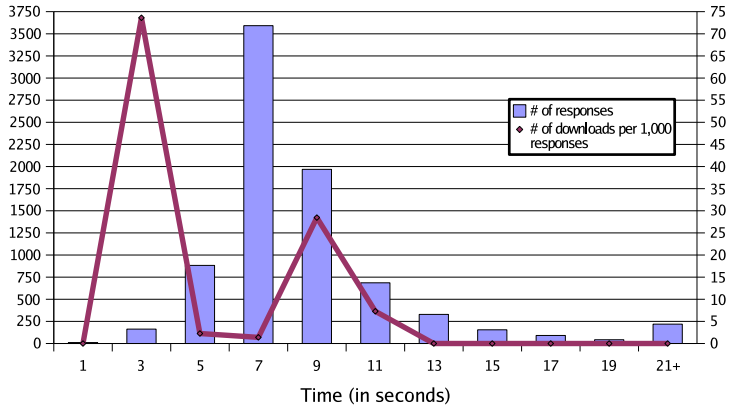


Figure 6: Query response time versus downloads (from the Overture experiment).

situations, our gateway system addresses information needs of WWW and P2P users, and enables otherwise Web-focused organizations to provide information and/or services to P2P networks.

In addition, we have provided evidence that:

- Gnutella users are interested in retrieving content available on the Web;
- When downloading through our system, users did not express a strong preference for particular sizes of files; and,
- Gnutella queries have a natural lifetime of approximately 12 seconds, after which results are ignored, and generally that faster responses are more likely to be useful.

In the future, we hope to better recognize certain request types, and as a result, be able to better direct the queries (similar to what has been suggested [5] for meta-search engines [13]). For example, when we receive a search for an mp3 file, we might choose (or learn) to direct it to an mp3-specific search engine, such as AudioGalaxy[14] instead of (or in addition to) a generic search engine. Elsewhere, we have demonstrated that caching the results of queries for short periods will increase the scalability of our system, allowing us to answer more queries without increasing the back-end resources required to handle a larger query load [2].

At present, we expect that query generation is likely to be self-selecting, and thus reflective of the types of content currently believed to be available on any particular peer-to-peer network. Thus, while we have demonstrated usage of our gateway, we expect that some users were likely disappointed with the

[14] http://www.audiogalaxy.com/

content — while the content includes the search terms, they may have different interpretations. Web search has long had to deal with this issue of polysemy, but the narrower file-sharing domain may not have experienced the problem to this extent.

Finally, while this work has focused on the development of a gateway from Gnutella to the Web, our approach is not limited to it. Any peer-to-peer network that permits searching across multiple, unaffiliated servers would also be a target. However, in some cases, tight coordination with the network designer would be required to operate correctly within an otherwise closed network.

### Acknowledgments

This report is based in part upon work supported by the National Science Foundation under Grant Number ANI-9903052.

## References

[1] E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), Oct. 2000. http://www.firstmonday.org/issues/issue5_10/adar/.

[2] B. D. Davison and W. Zhang. Searching the Web and more — a juxtaposition of online search traces. Technical Report LU-CSE-05-005, Dept. of Computer Science and Engineering, Lehigh University, 2005.

[3] B. D. Davison, W. Zhang, and B. Wu. Lessons from a Gnutella-Web gateway. In *Poster Proc. of the 13th Int'l World Wide Web Conference*, New York City, May 2004.

[4] A. Emtage and P. Deutsch. Archie – an electronic directory service for the Internet. In *Proceedings of the Winter Conference*, pages 93–110. Usenix Association, Jan. 1992.

[5] A. Howe and D. Dreilinger. Savvysearch: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2), 1997.

[6] G. Kan. Gnutella. In A. Oram, editor, *Peer to Peer: Harnessing the Benefits of Disruptive Technologies*, chapter 8, pages 94–122. O'Reilly, Sebastopol, CA, Mar. 2001.

[7] T. Klingberg and R. Manfredi. Gnutella 0.6, June 2002. http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html.

[8] Lime Wire, LLC. Rolling host count, Nov. 2003. Available from http://www.limewire.com/english/content/netsize.shtml.

[9] H. Niksic. GNU Wget – the noninteractive downloading utility, 2002. Available from http://www.gnu.org/software/wget/wget.html.

[10] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing*, 6(1), 2002.

[11] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[12] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[13] E. Selberg and O. Etzioni. Multi-service search and comparison using the MetaCrawler. In *Proceedings of the Fourth International World Wide Web Conference*, Boston, MA, Dec. 1995.

[14] K. C. Sia, C. H. Ng, C. H. Chan, S. K. Chan, and L. Y. Ho. Bridging the P2P and WWW divide with DISCOVIR – DIStributed COntent-based Visual Information Retrieval. In *Poster Proc. of the* $12^{th}$ *Int'l WWW Conference*, Budapest, May 2003.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[16] B. Wiley. Interoperability through gateways. In A. Oram, editor, *Peer to Peer: Harnessing the Benefits of Disruptive Technologies*, chapter 19. O'Reilly, Sebastopol, CA, Mar. 2001.