Monitoring The Gnutella Network

Stephan Brumme
Hasso-Plattner-Institute for Software Systems Engineering, Potsdam, Germany
currently at the University of Technology, Sydney, Australia

Abstract In this paper, I describe an intelligent agent crawling the Gnutella peer-to-peer network in order to detect clients in a certain IP range. This tool is helpful for administrator trying to avoid wasting bandwidth due to peer-to-peer file sharing programs. Moreover, almost all files shared over the internet are illegal – like non-licensed MP3 music – which can lead to potential law suits. In the past years, security leaks are often found in various server programs. Peer-to-peer network established as one of the major sources for internet viruses and worms. On the following pages, I will evaluate the opportunities and challenges of monitoring the Gnutella network from the technical, commercial and social point of view. I describe the implementation of GnutellaWatch which is a easy-to-use Java program suitable even for personnel without a broad understanding of network technology.

1 Introduction

Peer-to-peer networks, often abbreviated P2P, gained a lot of popularity over the past years. In the fall of 1999, the program Napster [URL:Napster] became available and immediately started a worldwide rush for file sharing. Originally written by a just single person, it turned out to be one of the success stories of the Internet boom. Its massive popularity reaches its peak in February 2001 when about 13.6 million user ran Napster on their computers.

The idea of file sharing is to set up a virtual network over the internet which can be crawled by a dedicated search engine. Once a file is found, it can be transferred from one user to another user, hence the name peer-to-peer. The major difference to classical client-server models like the file transfer protocol (FTP) is that there are no dedicated servers anymore. Each client now acts as a server, too.

Typical peer-to-peer network are quite vivid and change their structure all the time. Well-liked files are often available from multiple sources. In a perfect scenario, the download speed is only limited by the downloading node. However, this is almost never true but usually peer-to-peer networks under heavy load outperform web server under heavy load while allowing more users, too.

Especially freshly released downloads, like the major Linux distribution are usually handled better by peer-to-peer networks than by traditional architectures. For example, the latest Knoppix 3.4 release, [URL:Knoppix] a stunning 700 MByte giant, could only be distributed by file sharing tools in the first days since all mirrors worldwide were poorly available. The requested download bandwidth could not be satisfied by more than 50 servers, some of them even equipped with sophisticated load balancers.

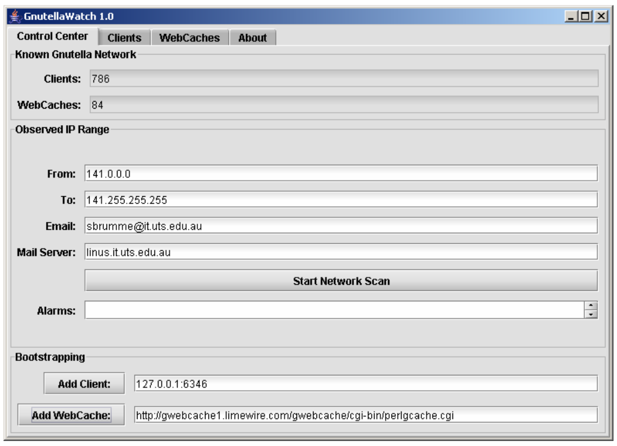
There is one major drawback of peer-to-peer networks: a very high percentage of the shared files infringe copyright restrictions. The music industry reported impressive losses due to MP3 sharing which caused declined CD sales.

Gnutella (spoken with a silent “g”) was released on March 14, 2000 by Justin Frankel, the main programmer of the famous WinAmp MP3 player [URL:Winamp]. Even though his company Nullsoft immediately closed down the download site, the program spread fast among the internet community. Gnutella is truly decentralized since the included search engine does not rely on dedicated servers. This is important since Napster was sued because the company offered dedicated search servers. In the end, Napster was closed down in July 2001.

Gnutella bases on a lean network protocol quite similar to the Hypertext Transfer Protocol (HTTP). The nodes share IP addresses of parts of the networks and satisfy/forward search queries. A node can be set up within a few seconds on almost every computer. The configuration is very flexible and remarkably helps to prevent the program from being detected by common network scan tools.

In the following chapters I will shortly describe the Gnutella protocol and how a program can hook

This paper was written as part of the lecture *Intelligent Agents* at the University of Technology, Sydney, in 2004. The author can be contacted via email:
stephan.brumme@hpi.uni-potsdam.de,
stephan.brumme@uts.edu.au, or
info@stephan-brumme.com



GnutellaWatch 1.0

Control Center | Clients | WebCaches | About

Known Gnutella Network

Clients:	786
WebCaches:	84

Observed IP Range

From:	141.0.0.0
To:	141.255.255.255
Email:	sbrumme@it.uts.edu.au
Mail Server:	linus.it.uts.edu.au

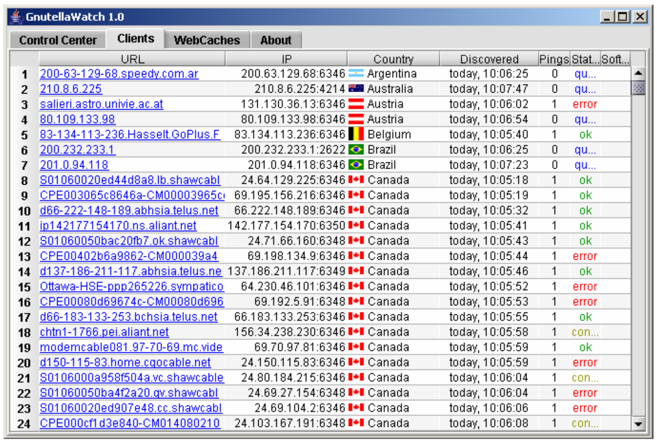
Start Network Scan

Alarms:

Bootstrapping

Add Client:	127.0.0.1:6346
Add WebCache:	http://gwebcache1.limewire.com/gwebcache/cgi-bin/perlgcache.cgi

Figure 1: Control Centre



GnutellaWatch 1.0

Control Center | Clients | WebCaches | About

	URL	IP	Country	Discovered	Pings	Stat...	Soft...
1	200-63-129-68.speedy.com.ar	200.63.129.68:6346	Argentina	today, 10:06:25	0	qu...	
2	210.8.6.225	210.8.6.225:4214	Australia	today, 10:07:47	0	qu...	
3	salieri.astro.univie.ac.at	131.130.36.13:6346	Austria	today, 10:06:02	1	error	
4	80.109.133.98	80.109.133.98:6346	Austria	today, 10:06:54	0	qu...	
5	83-134-113-236.Hasselt.GoPlus.F	83.134.113.236:6346	Belgium	today, 10:05:40	1	ok	
6	200.232.233.1	200.232.233.1:2622	Brazil	today, 10:06:25	0	qu...	
7	201.0.94.118	201.0.94.118:6346	Brazil	today, 10:07:23	0	qu...	
8	S01060020ed44d8a8.lb.shawcabl	24.64.129.225:6346	Canada	today, 10:05:18	1	ok	
9	CPE003065c8646a-CM00003965c	69.195.156.216:6346	Canada	today, 10:05:19	1	ok	
10	d66-222-148-189.abhsia.telus.net	66.222.148.189:6346	Canada	today, 10:05:32	1	ok	
11	ip142177154170.ns.aliant.net	142.177.154.170:6350	Canada	today, 10:05:41	1	ok	
12	S01060050bac20fb7.ok.shawcabl	24.71.66.160:6348	Canada	today, 10:05:43	1	ok	
13	CPE00402b6a9862-CM000039a4	69.198.134.9:6346	Canada	today, 10:05:44	1	error	
14	d137-186-211-117.abhsia.telus.ne	137.186.211.117:6349	Canada	today, 10:05:46	1	ok	
15	Ottawa-HSE-ppp265226.sympatico	64.230.46.101:6346	Canada	today, 10:05:52	1	error	
16	CPE00080d69674c-CM00080d696	69.192.5.91:6348	Canada	today, 10:05:53	1	error	
17	d66-183-133-253.bchsia.telus.net	66.183.133.253:6346	Canada	today, 10:05:55	1	ok	
18	chtn1-1766.pei.aliant.net	156.34.238.230:6346	Canada	today, 10:05:58	1	con...	
19	modemcable081.97-70-69.mc.vide	69.70.97.81:6346	Canada	today, 10:05:59	1	ok	
20	d150-115-83.home.cgocable.net	24.150.115.83:6346	Canada	today, 10:05:59	1	error	
21	S0106000a958f504a.vc.shawcable	24.80.184.215:6346	Canada	today, 10:06:04	1	con...	
22	S01060050ba4f2a20.gv.shawcabl	24.69.27.154:6348	Canada	today, 10:06:04	1	error	
23	S01060020ed907e48.cc.shawcabl	24.69.104.2:6346	Canada	today, 10:06:06	1	error	
24	CPE000cf1d3e840-CM014080210	24.103.167.191:6348	Canada	today, 10:06:08	1	con...	

Figure 2: Detected clients

itself into the network. Thus one is able collect many IP addresses of various active clients. I will explain how so-called web caches help to enter the network, too.

Depending on the country, Internet Service Providers may be held responsible for not preventing copyright infringements as they are done by almost every file sharing user. In addition, most companies have to pay a lot of money for the used network bandwidth which is drastically increased by peer-to-peer networks.

2 GnutellaWatch

2.1 Requirements

Before going into the details of the Gnutella network, I give a short overview of the intelligent agent I developed. I always had a user in mind who does not necessarily possess the knowledge of a network administrator. He does not need to know the details of protocols. A limited understanding of the way how the internet works is sufficient.

GnutellaWatch runs on every computer supporting Java 1.4 [URL:Java]. Without any modifications to the source code or recompilation, the same byte code can be executed on Windows machines as well on the vast variety of UNIX based operating systems.

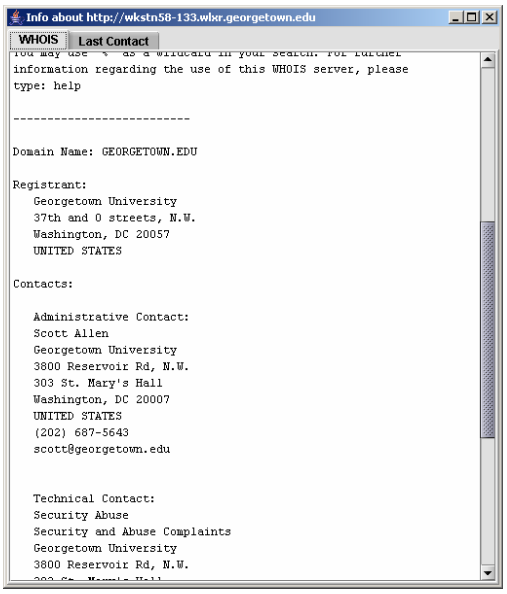
2.2 Usage

After starting the program, the user will see the Control Centre. Here he can start the Bootstrapping process that is required to join the Gnutella network. Furthermore, the user has the ability to scan a certain range of IP addresses (for example the network of his company) and request detailed e-mails for each detected Gnutella client.

Some runtime statistics show the size of the crawled network. My program is pretty fast and allows even on a low-end 100 KBit connection to retrieve about 800 unique IPs in just under two minutes.

Two tabs called "Clients" and "Webcaches" offer an in-depth examination of the known network. Based on the IP, my program determines the country of the remote nodes and performs a domain lookup to get a description of the originating network.

Very interesting is a feature that retrieves the corresponding WhoIS database entries when double-clicked on an table row. An entry exists for each domain worldwide and contains a lot of information about the administrator: usually including his e-mail address and often even his office telephone numbers. If you found a suspicious IP, why not give the responsible administrator a call ?



Info about http://wkstn58-133.wlxr.georgetown.edu

WHOIS | Last Contact

```
You may use "%" as a wildcard in your search. For further
information regarding the use of this WHOIS server, please
type: help

--------------------------

Domain Name: GEORGETOWN.EDU

Registrant:
   Georgetown University
   37th and O streets, N.W.
   Washington, DC 20057
   UNITED STATES

Contacts:

   Administrative Contact:
   Scott Allen
   Georgetown University
   3800 Reservoir Rd, N.W.
   303 St. Mary's Hall
   Washington, DC 20007
   UNITED STATES
   (202) 687-5643
   scott@georgetown.edu


   Technical Contact:
   Security Abuse
   Security and Abuse Complaints
   Georgetown University
   3800 Reservoir Rd, N.W.
   303 St. Mary's Hall
```

Figure 3: WhoIS information

3 Agent Technology

Even though they are some smart and useful applications based on peer-to-peer networks, they are now seen as a big danger because of copyright infringements. My program GnutellaWatch employs internet agent technology [Debenham04] to discover active Gnutella clients.

It intelligently choses between two underlying discovery protocols and can detect improper protocol implementations.

The environment of GnutellaWatch is quite huge: it is the whole internet ! My program has to carefully select interesting parts without harming innocent computers.

In order to find an active Gnutella client, one could write a brute-force program. A Gnutella client can potentially found on any free port of an IP. Since they are about 2^{16} ports defined, even a medium sized B-class subnet would require 2^{32} port scans. If that brute-force solution would be able to perform 100 port scans per second – which is very fast – it would need 1.5 years. My program cannot guarantee to find all Gnutella clients but is much faster and friendlier to its environment.

GnutellaWatch tries to fight the Gnutella network, not to support it. Therefore, it behaves passive to other clients and operates in "stealth mode". It only listens to current network activity without contributing to potentially illegal actions.

The ever-changing shape of the Gnutella network is responsible for inevitable errors. If a remote server was misinterpreted as a part of the Gnutella network, the program will stop from bothering it.

There is no need for configuration or ongoing supervision: the program consists of just two files that can be copied to a new location and run everywhere you find a Java Virtual Machine and internet access. Firewalls are not a major problem. Except for resources issues, GnutellaWatch could run forever without ever needing any interacting from the user. It can send its reports via e-mail. So it is possible to start GnutellaWatch on a computer you rarely access and only read all status reports to actively fight the Gnutella network.

My program is not exactly mobile but since of its as-simple-as-possible deployment approach I would like to call it "virtually mobile".

4 The Gnutella Protocol

4.1 Bootstrapping

Whenever a Gnutella client is started, it does not know which other nodes are currently available on the internet. Therefore the first step is to find at least one active node. This process is called bootstrapping.

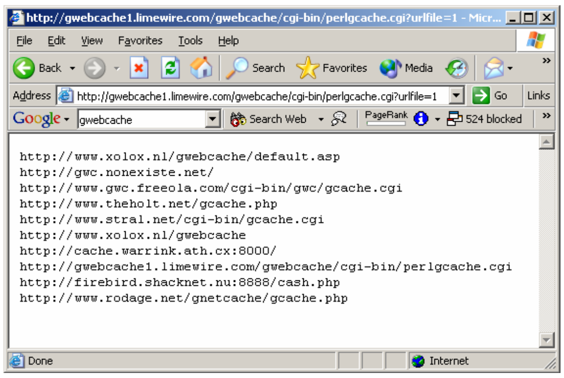
There are no centralized servers available to help during this process. If a client would perform a dumb trial-and-error bootstrapping, for example pinging all computers whether they run Gnutella or not, then this would infer an enormous waste of resources.

Two basic bootstrapping algorithms established: a client should store all known nodes of the last session and on the next start ping all these IPs. Since file sharing tools often run in the background and modern computers are almost never switched off, there is a good chance to find a valid node. Moreover, if incidentally the computer connected via an IP changed (quite common for dial-up internet access) but the new computer runs Gnutella, too, then bootstrapping works as well since it does not matter which Gnutella client is found. They all behave the same.

The second bootstrapping algorithm is more reliable and works with so-called "web caches" [URL:GWebCache]. They are ordinary web sites powered by specialized PHP, ASP or Perl scripts. All they do is to return some random valid IPs of clients they recently registered. Web sites change every day – therefore the web caches know each other and provide URLs of other web caches, too. A good Gnutella client stores all web caches of the last session in the same manner it does for IPs.

Basically, web caches are a second network layered above the Gnutella network. They are pretty generic since they just know some IPs and URLs. It should be fairly easy to set up web caches for one of the other major file sharing networks like Kazaa [URL:Kazaa] which was developed by the Sydney-based company Sharman Networks.

Gnutella web caches follow the *gwebcache* standard [URL:GWebCache]. This standard is not officially recognized but implemented by a dozen of open source projects. A search for gwebcache on Google returns about 1,000 links which either point to a web cache or to a developer site. During my tests I consistently found more than 50 active web caches.


http://gwebcache1.limewire.com/gwebcache/cgi-bin/perlgcache.cgi?urlfile=1 - Micr...
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://gwebcache1.limewire.com/gwebcache/cgi-bin/perlgcache.cgi?urlfile=1
Go
Links
Google
gwebcache
Search Web
PageRank
524 blocked
Done
Internet

```
http://www.xolox.nl/gwebcache/default.asp
http://gwc.nonexiste.net/
http://www.gwc.freeola.com/cgi-bin/gwc/gcache.cgi
http://www.theholt.net/gcache.php
http://www.stral.net/cgi-bin/gcache.cgi
http://www.xolox.nl/gwebcache
http://cache.warrink.ath.cx:8000/
http://gwebcache1.limewire.com/gwebcache/cgi-bin/perlgcache.cgi
http://firebird.shacknet.nu:8888/cash.php
http://www.rodage.net/gnetcache/gcache.php
```

Figure 4: A browser retrieving a web cache

An ordinary web browser is sufficient to view such a web cache as is shown in figure 4.

4.2 Friend Of A Friend

Once we got a connection to at least one node of the Gnutella network, we can ask that node to give us the addresses of some of the nodes it is connected to. Then we ask these new nodes again for the same information and so on. This algorithm is sometimes called "Friend of a friend" and proves to gain a pretty fast access to a huge user base.

An example: if a clients know 7 other nodes, these in turn known 7 new nodes then we already know 7+7*7=56 nodes. If we ask the most recent 7*7 nodes we will probably get 7*7*7=343 nodes. After only six steps we have identified about 100,000 nodes !

Unfortunately, reality always differs from this theoretical example. Even though most nodes can actually return 7 other nodes, many of the nodes overlap. A node X we got by asking node Y will certainly return node X. It is not unusual to get invalid IP – some nodes even reject to deliver addresses of their friends.

4.3 Gnutella Commands

Message ID	Type	Time To Live	Hops	Data Length	Data
16 bytes	1 byte	1 byte	1 byte	4 bytes	varying

Figure 5: Message Architecture

The communication protocol of Gnutella is very much alike the HTTP protocol. Indeed, the download of files is pure HTTP/1.1 while pings and queries (explained later) differ a bit.

I will describe only the more modern Gnutella 0.6 protocol [URL:GnutellaRFC] understood by more than 99% of all Gnutella clients. Don't be confused by Gnutella2 which you may find on the internet. That protocol is completely different and the whole Gnutella community is upset about the misuse of the name "Gnutella" for marketing purposes of an incompatible program [URL:Shareaza].

4.3.1 *Hand-Shaking*

To establish a connection, the requesting node has to send:

```
GNUTELLA CONNECT/0.6<cr><lf>
```

When the attempt to connect was successful, a response according to this pattern is generated:

```
GNUTELLA/0.6 200 <string><cr><lf>
```

String is usually two letters: OK.

By now, nodes add several header fields to these one-line commands. A more typical hand-shake therefore looks like this:

[Request]

```
GNUTELLA CONNECT/0.6<cr><lf>
User-Agent: GnutellaWatch/1.0<cr><lf>
Ultra-Peer: False<cr><lf>
Pong-Caching: 0.1<cr><lf>
```

[Response]

```
GNUTELLA/0.6 200 OK<cr><lf>
User-Agent: LimeWire/1.0<cr><lf>
X-Try: 141.2.89.3:6436,
    212.43.98.71:6436<cr><lf>
```

As one can see, most nodes expose their name and some friends during the hand-shake.

4.3.2 *Standard Message Architecture*

Now that we got an established connection, we can send binary messages built according to figure 5.

The message ID should be globally unique. There are many libraries available to generate such as number often referred to as GUID.

The *type* can be:

- *ping* (0x00),
- *pong* (0x01),
- *bye* (0x02),
- *push* (0x40),
- *query* (0x80),
- *query hit* (0x81)

4.3.3 *Ping Message*

This message only checks whether a client is still connected to the network. It contains no data and should not be confused with the `ping` command of most operating systems.

4.3.4 Pong Message

This message is only sent as a response to a ping message. It carries some information about the number and size of shared files.

4.3.5 Bye Message

This message indicates that the client will soon leave the network. It carries no further data and can be omitted.

4.3.6 Push Message

This message sends (pushes) a file. It is somehow redundant since file transfer usually bases on HTTP. Most clients omit this message.

4.3.7 Query

This message asks for a certain file. Beside the file name, a query also contains some transfer speed restrictions. For example, it may ask for at least 10 KByte/sec. The message size is in most cases below 200 bytes.

4.3.8 Query Hit

This message returns one or more sources for a file requested by a query. Most important, it carries the IP addresses of all source node, their connection speed and a file identifier. This identifier is used when actually requesting the file via HTTP.

4.4 Problems

4.4.1 Bandwidth Issues

The Gnutella heavily employs pings and queries. Especially queries can cause a huge amount of traffic: if a node does have a file that was requested, it forwards the queries to all of his friends. For rarely found files there is an exponential growth according to the nodes the query passes – clearly a bottleneck.

Each message therefore contains a *time to live* field. It is decreased by one every time a query is forwarded. Once it reaches zero, the query is discarded and removed from the network. A typical initial value for *time to live* is 7.

In addition to decreasing the *time to live*, a second header field called *hops* is increased by one and therefore gives us some knowledge about the nodes the message already passed. The idea behind this concept is to prevent clients from using too high initial *time to live* values (which may slightly increase the query hit probability but causes significantly more traffic): the sum of *time to live* and *hops* is always constant. If this sum is higher than a certain threshold then a node is allowed to decrease the time to live until the sum is in a valid range.

Messages are quite short. Besides some protocol extensions, only *pong*, *query* and *query hit* contain some data. In most cases, their size is below 300 bytes. If the size exceeds 4 Kbytes, a node can remove the message.

4.4.2 Too Many Connections

Gnutella is “connection-greedy”. While implementing my program, I got a lot of “connection refused” responses in the hand-shake.

I still do not know whether the nodes actually suffer from too many connections or only want to save bandwidth and protect their privacy.

4.4.3 Trusting Nodes

One can never rely on the messages transferred over the Gnutella network. Some commercial clients try to inject spam in the network.

It was told that a few modified nodes tried to exploit security holes in some open-source clients. Since the code of many Gnutella clients is publicly available, found leaks can lead to severe damages:

- remote clients can gain local root access
- insecure clients may be misused to send spam or, even worse, worms
- keylogger etc. can reveal information that should have kept secret
- … and many more !

Right now, there is no solution for these problems because of the simple text-based protocol that works without any encryption. Nevertheless, the big majority of clients can be trusted.

4.4.4 Invalid nodes

The *bye* message is not always properly used. Some clients leave the networks without notifying their friends. The *ping* message is sent periodically to update the availability of all friends but cannot be sent too often since that would incur a substantial increase of network traffic.

In consequence, a node only knows which of its friends were online some time ago. When asked by a new node for its friends, a node may returns addresses of friends that are no longer online.

Depending on the timespan between two *ping*s, I observed a medium up to high percentage of invalid addresses.

The problem is even worse for web caches. They usually allow updates once an hour to conserve bandwidth but many clients are online only for a couple of minutes.

4.5 Advanced Topics

4.5.1 File Swarming

A typical audio CD compressed in the widespread MP3 file format occupies 60 MByte while an average DivX encoded movie requires up to 1 GByte.

Downloading a file just from a single source would limit the download speed to the upload speed of the source. Today, asymmetric DSL connections have a low upload speed which is not practical.

If we rely on a download from one source than we have to ensure that it is online for the whole download. Especially for large files (often several Mbytes) this cannot be guaranteed.

File swarming helps to eliminate these problems: part of the HTTP protocol is a partial download. Gnutella almost always requests certain blocks and joins them locally. It is no problem to download the last few bytes of a file first and the first bytes of that file last. Parallel downloads from multiple sources are handled without major worries.

4.5.2 Extensions

Clients like LimeWire [URL:LimeWire] or BearShare [URL:BearShare] introduced new extensions to the protocol to enhance bootstrapping and queries. Not all extensions are widely accepted but some promise real improvements, though: high-speed connected Ultrapeers bundle messages, clients cache queries and file magnets provide a web interface fully replacing the need of a Gnutella client.

Up to now, there is no new standard and even Gnutella version 0.6, as used in this paper, is still a draft [URL:GnutellaRFC].

5 Foundations of GnutellaWatch

5.1 Bootstrapping

I wrote code to access web caches as well as clients in order to perform bootstrapping.

A web cache returns a text list of URLs of the web caches it knows, one per line. The same holds true for client IPs but they have to be fully qualified with a port number, usually 6436.

To get the web cache URLs, the web cache has to be called with a parameter *urlfile=1*. In figure 4, the full URL to retrieve the URLs of known web caches was:
http://gwebcache1.limewire.com/gwebcache/cgi-bin/perlgcache.cgi?urlfile=1 (URL has been split into two lines to fit the layout of this paper). Asking another web cache for a list of clients is as simple as this: http://www.xolox.nl/gwebcache/?hostfile=1.

If something goes wrong, the web cache has respond ERROR. I often observed ordinary HTML pages so I verify each line whether it is a standard-conforming URL and discard all lines that fail this simple test.

In my Java code, the classes `HttpURLConnection` and `URL` are basically all I need to perform web cache based boot strapping.

The class is called `WebCacheSpider` and processes one web cache at a time returning all new web caches and nodes found.

5.2 Requesting Node IPs

In the last chapter on bootstrapping I already explained how to get client or node IP from a web cache.

While showing the process of hand-shaking I gave an example of inserted special header fields. The fields `X-Try` and `X-Try-Ultrapeer` are supported by most Gnutella clients and give a comma-separated list of all friends of a node.

The dedicated Java class serving that purpose is called `ClientSpider` and processes one Gnutella client at a time returning all new nodes found - exactly the same way as `WebCacheSpider`.

5.3 Further IP information

5.3.1 Reverse DNS Lookup

Modern DNS servers allow to resolve an IP adderss and return the main textual representation: a URL. The process may give invalid results for shared hosting but usually works quite reliable.

Java's nice internet library reduce the code to make a reverse DNS lookup to the shortest solution possible: `InetAddress`' method `getHostName` gives us the desired URL as a String.

5.3.2 GeoIP

Sometimes a reverse DNS lookup fails. Furthermore, my program generates a massive DNS traffic since it sometimes sends 10 queries per seconds. As stated before, for shared hosting there is no unique reverse DNS lookup.

A faster and locally available solution is GeoIP developed by MaxMind LLC [URL:GeoIP]. Even though its full version costs some money, there is a basic database and Java source code available for free.

The GeoIP library gives me information about the IP's country of origin. This is especially worthy for international domains like *com*, *net* or *org*.

5.3.3 WhoIS

The WhoIS protocol was designed in the early days of the internet [Whois85] to retrieve information about the administrator of a domain.

Since he actually owns the domain, he is responsible for the whole contents of his publicly accessible server.

Depending on the country, the underlying database contains the postal address, the e-mail address and the telephone number of the administrator. An example is given in figure 3.

The protocol itself is very easy: just send the domain you would like to get information about to a responsible WhoIS server on port 43. Then the server will respond with the desired contents. Unfortunately, no true standard exists for the format of the result.

There are two pitfalls: the domain name has to be reversed. For example www.stephan-brumme.com has to be changed to com.stephan-brumme.www. The other problem: WhoIS is generally only available for second-level domain. The exemplary domain has to be reduced to com.stephan-brumme. Some exceptions are countries like Australia that have predefined second-level domains like edu.au. In these countries, WhoIS will work for third-level domains like uts.edu.au (or reversed: au.edu.uts) as well.

There are various ways to determine the responsible WhoIS server. Often they can be found at whois.nic.[insert country domain here]. Again, no standard defines this, it is only a common observation. I decided to send all (!) my WhoIS requests to whois.geektools.com [URL:GeekTools]. They run a frequently updated database, automatically select the responsible WhoIS server and send me all I need. Since this service is for free, geektools.com restricts the service to 50 requests per IP per day.

I encapsulated the WhoIS request in the class `WhoIs`. It is straightforward `Socket` based code and thanks to Java's internet functionality short in length.

6 GnutellaWatch Architecture

6.1 Third Party Libraries

An agent should be small in size since it is typically run only for a short timespan. Easy and seamless integration in new environment is very important for agents.

I decided implement the involved agent technology by writing the code on my own. This way I best learn how to construct an agent and get the best possible understanding of the main issues.

Moreover, third party libraries not always fit the needs and have to be adapted or wrapped. This leads to an exhausting code bloat that I tried to avoid.

Nevertheless, I used the GeoIP [URL:GeoIP] library to lookup the country of an IP. The code really worked "as is" and perfectly suited my needs: it offers all the features that I need – not more.

I do not like the look of Java applications. They somehow do not fit into the modern user interfaces of Windows, MacOS Aqua or KDE. A more pleasant user interface is provided by the open source Kunststoff project [URL:Kunststoff]. They add a layer on top of Java's Swing Metal look'n'feel and notably improve the overall experience. In my past semesters I had some lectures on the design of user interfaces and discovered how seldom programmers have the common user in mind. Especially computer scientists tend to forget about reasonable usability. Fortunately, the internet and its web designers revolutionized the way how we treat human machine interaction.

6.2 Multithreading

Nowadays' internet program are usually limited by the offered bandwidth and the speed of remote computer. The locally available CPU power becomes more and more negligible; however memory is still a potential bottleneck.

Most web caches and nodes need between 0.5 and 2 seconds to respond to my requests. This may be caused by my current location: Australia is quite far away from the big file sharing networks in Europe and America.

It is not feasible to process only one request at a time. Instead, multiple requests can be performed to significantly increase the performance of Gnutella-Watch. Thus, I more efficiently utilize available resources and, most important, save much time [Christopher00, Lewis00].

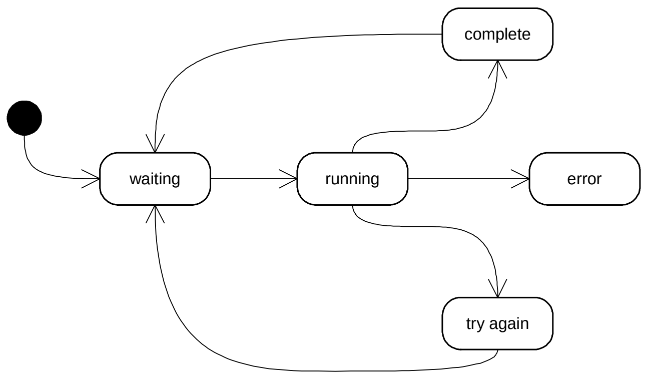


Figure 7: Queue State Chart

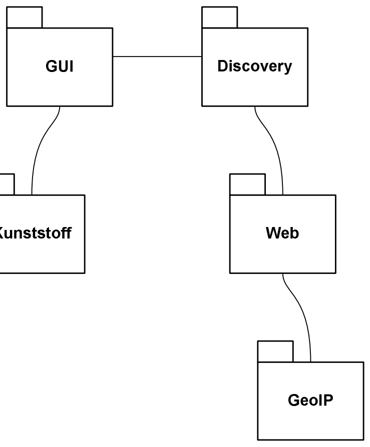


Figure 8: Packages

Multithreading always implies synchronization issues. I handle them by so-called managers. For all object of the class `WebCacheSpider`, there is a single `WebCacheManager`. For all objects of the class `ClientSpider`, there is a single `ClientManager` [Heaton02].

6.3 Spiders And Manager

The `WebCacheSpider` and the `ClientSpider` basically differ in the protocol they understand. Because of many shared features, both implement the interface `ISpider`. A similar relationship exists between `WebCacheManager` and `ClientManager`: they derive from `InfoManager`.

The job of a manager is to spawn multiple spiders and keep track of the workload by supervising a system of queues (see below).

Almost all methods of the managers are declared synchronized to prevent one spider interrupting another while performing queue updates. If I would omit it, the queues may be corrupted which could crash the whole program.

6.4 Queues

There a five states for each address (no matter whether web cache address or client IP): waiting, running, complete, error and try-again [Heaton02].

Initially, an address represented by the `IInfo` interface is set to be in the *waiting* state. This is done by pushing it to the waiting queue. When a spider is asking for work, it removes one address from the waiting queues and declares it as *running* by adding it to the running queue. When done, the address is, depending on the result of the spider, either moved to the *complete*, *error* or *try-again* state.

I called these data structures queues because a new entry is always added to the end while removing always occurs at the head. In computer science terminology, these queues are strictly speaking FIFO lists.

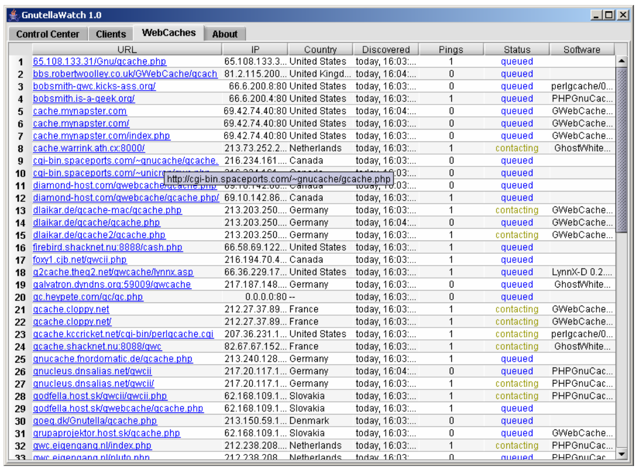
For convenience reasons, a separate list keeps track of all `IInfo` objects. This simplifies the GUI programming a bit.

6.5 Namespaces

I like the concept of namespaces to organize a source code tree and avoid name clashes. In the figure about, you see the main namespaces. The packages (Java's name for namespaces) are only loosely coupled. GeoIP and Kunststoff are third-party packages and used "as is".

6.6 User Interface

An agent is defined as an entity that does something on behalf of someone. GnutellaWatch spiders the GnutellaWatch on behalf of an administrator, that means a human being.



	URL	IP	Country	Discovered	Pings	Status	Software
1	65.108.133.31/Gnu/gcache.php	65.108.133.3...	United States	today, 16:03:...	1	queued	
2	bbs.robertwoolley.co.uk/GWebCache/gcach	81.2.115.200...	United Kingd...	today, 16:04:...	0	queued	
3	bobsmith-gwc.kicks-ass.org/	66.6.200.8:80	United States	today, 16:03:...	0	queued	perlgcache/0...
4	bobsmith.is-a-geek.org/	66.6.200.4:80	United States	today, 16:03:...	1	queued	PHPGnuCac...
5	cache.mynapster.com	69.42.74.40:80	United States	today, 16:04:...	0	queued	GWebCache...
6	cache.mynapster.com/	69.42.74.40:80	United States	today, 16:03:...	0	queued	GWebCache...
7	cache.mynapster.com/index.php	69.42.74.40:80	United States	today, 16:03:...	0	queued	GWebCache...
8	cache.warrink.ath.cx:8000/	213.73.252.2...	Netherlands	today, 16:03:...	1	contacting	GhostWhite...
9	cgi-bin.spaceports.com/~gnucache/gcache.	216.234.161....	Canada	today, 16:03:...	0	queued	
10	cgi-bin.spaceports.com/~unicr[illegible]	216.234.161...	Canada	today, 16:03:...	0	queued	
11	diamond-host.com/gwebcache/gcache.php	69.10.142.86...	Canada	today, 16:03:...	0	queued	
12	diamond-host.com/gwebcache/gcache.php/	69.10.142.86...	Canada	today, 16:03:...	1	queued	
13	dlaikar.de/gcache-mac/gcache.php	213.203.250....	Germany	today, 16:03:...	1	contacting	GWebCache...
14	dlaikar.de/gcache/gcache.php	213.203.250....	Germany	today, 16:04:...	0	queued	GWebCache...
15	dlaikar.de/gcache2/gcache.php	213.203.250....	Germany	today, 16:03:...	1	contacting	GWebCache...
16	firebird.shacknet.nu:8888/cash.php	66.58.69.122...	United States	today, 16:03:...	1	queued	
17	foxy1.cjb.net/gwcii.php	216.194.70.4...	Canada	today, 16:03:...	1	queued	
18	g2cache.theg2.net/gwcache/lynnx.asp	66.36.229.17...	United States	today, 16:03:...	1	queued	LynnX-D 0.2....
19	galvatron.dyndns.org:59009/gwcache	217.187.148....	Germany	today, 16:03:...	0	queued	GhostWhite...
20	gc.heypete.com/gc/gc.php	0.0.0.0:80	--	today, 16:03:...	0	queued	
21	gcache.cloppy.net	212.27.37.89...	France	today, 16:03:...	1	contacting	GWebCache...
22	gcache.cloppy.net/	212.27.37.89...	France	today, 16:03:...	1	contacting	GWebCache...
23	gcache.kccricket.net/cgi-bin/perlgcache.cgi	207.36.231.1...	United States	today, 16:03:...	1	contacting	perlgcache/0...
24	gcache.shacknet.nu:8088/gwc	82.67.67.152...	France	today, 16:03:...	1	contacting	GhostWhite...
25	gnucache.fnordomatic.de/gcache.php	213.240.128....	Germany	today, 16:03:...	1	queued	
26	gnucleus.dnsalias.net/gwcii	217.20.117.1...	Germany	today, 16:04:...	0	queued	PHPGnuCac...
27	gnucleus.dnsalias.net/gwcii/	217.20.117.1...	Germany	today, 16:03:...	1	contacting	PHPGnuCac...
28	godfella.host.sk/gwcii/gwcii.php	62.168.109.1...	Slovakia	today, 16:03:...	1	contacting	PHPGnuCac...
29	godfella.host.sk/gwebcache/gcache.php	62.168.109.1...	Slovakia	today, 16:03:...	1	queued	
30	goeg.dk/Gnutella/gcache.php	213.150.59.1...	Denmark	today, 16:03:...	0	queued	
31	grupaprojektor.host.sk/gcache.php	62.168.109.1...	Slovakia	today, 16:03:...	0	queued	GWebCache...
32	gwc.eigengang.nl/index.php	212.238.208....	Netherlands	today, 16:03:...	1	contacting	PHPGnuCac...
33	gwc.eigengang.nl/pluto.php	212.238.208....	Netherlands	today, 16:03:...	0	queued	PHPGnuCac...

Figure 6: An Enhanced User Interface

All interaction between humans and computers has to be adapted to the needs and requirements of the human not the computer. In the early ages of computer science, often the inverse was true.

The change over the last years is best symbolized by the move from command-line interface to graphical user interfaces.

But not all graphical user interfaces are good interfaces: colours should be used more carefully. Sometimes, more colours enhance the user interface; sometimes you should reduce the number of involved colours.

Working with large tables can be annoying: when reading a row from left to right, many users incidentally slip. A subtle background differentiating odd from even row helps a lot [Goldstein01].

Users often want immediate feedback or try to explode by double-clicking something interesting. My tables support this and even show small tooltips while the mouse pointer is hovering above a certain row. Last but not least, all columns can be sorting in an ascending order [Linden02].

6.7 Software Quality

Excellent support of exceptions is the most outstanding feature of Java. I catch them to recover from “exceptional” situations and bring the program back to a normal state.

The nature of the internet implies many exceptions: URLs may be spelt incorrectly, connections may suddenly be interrupted and many more.

It is hard to understand the current state of a multithreaded program. While debugging GnutellaWatch I often had severe problems understanding the cause of some bugs. Therefore I learnt how to use the Java Logging Library – a major step in software quality since now I can trace the program paths.

Documentation is often underestimated. Especially for one-man-projects like GnutellaWatch one finds a lack of good documentation. To me, documentation is not only writing a paper (the one you are currently reading) but also source comments like JavaDoc and understandable graphical descriptions. UML diagrams often tend to be over-stressed. They can be quite useful but they have to be stripped down to the essential core – most UML evangelists do not realize this. Nevertheless, I have some UML diagrams to explain the basic relationships of GnutellaWatch. According to theories of cognition sciences, I try to show at most seven elements in a single diagram.

7 Future Improvements

Nobody is perfect. GnutellaWatch is a ready-to-use solution proudly stating “version 1.0”. This is true, is a fully functional release without any major problems.

GnutellaWatch is not perfect in a way that it ideally should be able to fulfil its job faster, more accurate and, most important, easier.

- One of the issues to be solved in a next release is persistency. Right now, GnutellaWatch does not save its current state to disc and requires a lot of redundant crawling each time. A XML serializer would be preferred.
- I did not implement code to actually send and receive Gnutella Messages – all I do is the hand-shake. Even though the hand-shake is sufficient to obtain IPs, evaluating the messages could help to analyse whether a user shares illegal contents over the internet.
- The usage of resources is not optimal. In addition to Java’s bad performance, GnutellaWatch may run quite slow even on fast computers.
- Traffic shaping could allow a better control about the network traffic generated by my program. This is important when GnutellaWatch is intended to run for a long term.
- Some constants like the number of threads are hard-coded. They should configurable in the next release.
- GnutellaWatch should be more personalized. This means that the user should be able to adapt the look’n’feel as well as the search options to his own needs.
- My program is unable to understand the recently proposed GWebCache2 protocol. It gives better performance and control for web caches.
- Basically, the protocol of competing file sharing networks like FastTrack (used in Kazaa), eDonkey and Kademlia are be spidered, too. Some of them are documented on the internet. My open architecture should allow implementing these protocols without major hassles.

Now I would like to talk about topics programmers often deny to mention:

- I completely underestimated the size of the project. Next time I need a better planning to avoid hectic work in the last hours before the final deadline.
- No source control system was used. Modern Java IDEs support a wide variety, like CVS and Subversion. They help to keep track of changes and bugs.
- There is no help system. A proper user interface must have a (context-sensitive) help system.

8 References

8.1 Literature

[Christoper00] Thomas W. Christopher, George K. Thiruvathukal, *High-Performance Java Platform Computing*, Sun Microsystems Press, Palo Alto, 2000

[Debenham04] John Debenham, Lecture *32530 Building Intelligent Agents*, held at the University of Technolgy, Sydney, autumn semester 2004

[Goldstein01] Mitch Goldstein, *Hardcore JFC*, Cambridge University Press, Cambridge, 2001

[Heaton02] Jeff Heaton, *Programming Spiders, Bots and Aggregators in Java*, Sybex, San Francisco, 2002

[Jones03] M. Tim Jones, *AI Application Programming*, Charles River Media, Hingham, 2003

[Lewis00] Bil Lewis, Daniel J. Berg, *Multithreaded Programming with Java Technology*, Sun Microsystems Press, Palo Alto, 2000

[Linden02] Peter van der Linden, *Just Java 2*, 5th edition, Sun Microsystems Press, Palo Alto, 2002

[Whois85] K. Harrenstien, M. Stahl, E. Feinler, *Nicname/Whois*, RFC 954, 1985

8.2 Online Material

[URL:BearShare]
www.bearshare.com

[URL:GeekTools]
www.geektools.com

[URL:GeoIP]
www.maxmind.com

[URL:GnutellaRFC]
http://rfc-gnutella.sourceforge.net

[URL:GWebCache]
www.gnucleus.com/gwebcache/

[URL:Java]
java.sun.com

[URL:Kazaa]
www.kazaa.com

[URL:Knoppix]
www.knoppix.net

[URL:Kunststoff]
www.incors.org

[URL:LimeWire]
www.limewire.com

[URL:Napster]
www.napster.com,
en.wikipedia.org/wiki/Napster

[URL:Shareaza]
www.shareaza.com

[URL:Winamp]
www.winamp.com