



www.archive.org
415.561.6767
415.840-0391 e-fax

Internet Archive
300 Funston Avenue
San Francisco, CA 94118

AFFIDAVIT OF CHRISTOPHER BUTLER

1. I am the Office Manager at the Internet Archive, located in San Francisco, California. I make this declaration of my own personal knowledge.

2. The Internet Archive is a website that provides access to a digital library of Internet sites and other cultural artifacts in digital form. Like a paper library, we provide free access to researchers, historians, scholars, and the general public. The Internet Archive has partnered with and receives support from various institutions, including the Library of Congress.

3. The Internet Archive has created a service known as the Wayback Machine. The Wayback Machine makes it possible to surf more than 400 billion pages stored in the Internet Archive's web archive. Visitors to the Wayback Machine can search archives by URL (i.e., a website address). If archived records for a URL are available, the visitor will be presented with a list of available dates. The visitor may select one of those dates, and then begin surfing on an archived version of the Web. The links on the archived files, when served by the Wayback Machine, point to other archived files (whether HTML pages or images). If a visitor clicks on a link on an archived page, the Wayback Machine will serve the archived file with the closest available date to the page upon which the link appeared and was clicked.

4. The archived data made viewable and browseable by the Wayback Machine is compiled using software programs known as crawlers, which surf the Web and automatically store copies of web files, preserving these files as they exist at the point of time of capture.

5. The Internet Archive assigns a URL on its site to the archived files in the format [http://web.archive.org/web/\[Year in yyyy\]\[Month in mm\]\[Day in dd\]\[Time code in hh:mm:ss\]/\[Archived URL\]](http://web.archive.org/web/[Year in yyyy][Month in mm][Day in dd][Time code in hh:mm:ss]/[Archived URL]). Thus, the Internet Archive URL <http://web.archive.org/web/19970126045828/http://www.archive.org/> would be the URL for the record of the Internet Archive home page HTML file (<http://www.archive.org/>) archived on January 26, 1997 at 4:58 a.m. and 28 seconds (1997/01/26 at 04:58:28). A web browser may be set such that a printout from it will display the URL of a web page in the printout's footer. The date assigned by the Internet Archive applies to the HTML file but not to image files linked therein. Thus images that appear on a page may not have been archived on the same date as the HTML file. Likewise, if a website is designed with "frames," the date assigned by the Internet Archive applies to the frameset as a whole, and not the individual pages within each frame.

6. Attached hereto as Exhibit A are true and accurate copies of printouts of the Internet Archive's records of the HTML files for the URLs and the dates specified in the footer of the printout.

7. As noted above in paragraph 5, the 14-digit timestamp included in an archived file's URL denotes the Internet Archive's record for the time of capture of that file in year, month, day, hours, minutes, and seconds, with the the last two digits in the timestamp indicating the seconds. For the following archived files, printouts for which are provided as part of Exhibit A, the Internet Archive's record of the year, month, day, hours, and minutes for the time of capture are reflected in the timestamps included in these files' URLs. However, in these specific instances a record for the seconds portion of the time of capture is absent from the timestamps, which all end in "00". For these specific files, the ending "00" is a result of this absence of a record for seconds and



should not be interpreted to reflect that the time of a capture was at 00 seconds (i.e., at the change to a new minute).

<https://web.archive.org/web/20001212195200/http://freenet.sourceforge.net/index.php?page=theoppr>

<https://web.archive.org/web/20010112014900/http://capnbry.dyndns.org/gnutella/gnutellatrans.html>

<https://web.archive.org/web/20010202024800/http://www.limewire.com/>

<https://web.archive.org/web/20010124044000/http://www.lucent.com/serviceprovider/imminet/>

8. Attached hereto as Exhibit B is a true and accurate electronic copy of the Internet Archive's records of the ps.gz file for the URL <http://freenet.sourceforge.net/icsi.ps.gz> and the date June, 2, 2001 UTC (with Wayback Machine URL: <https://web.archive.org/web/20010602211226/http://freenet.sourceforge.net/icsi.ps.gz>).

9. I declare under penalty of perjury that the foregoing is true and correct.

DATE: 9/16/15



Christopher Butler

CALIFORNIA JURAT

See Attached Document.

A notary public or other officer completing this certificate verifies only the identity of the individual who signed the document to which this certificate is attached, and not the truthfulness, accuracy, or validity of that document.

State of California
County of San Francisco

Subscribed and sworn to (or affirmed) before me on this

16th day of September, 2015, by

Christopher Butler,

proved to me on the basis of satisfactory evidence to be the person who appeared before me.



Signature: 

Exhibit A

[Home](#)

Welcome to gnutella

[INVITE OTHERS](#) | [HELP](#) | [FEEDBACK](#)
GUEST: [LOG IN](#) | [JOIN GROUP](#)[What's New](#)[Developer's Corner](#)[Help/Support](#)[Gnutella?](#)[Tutorial](#)[FAQ](#)[Software](#)[Third-Party](#)[Download](#)[Press](#)[Promote Gnutella](#)[About](#)[Site Map](#) [Community](#)[Discussion Board](#) [Resources](#)[File Center](#)[Links](#)**1743392**

Since April 10th, 2000

Gnutella's Home On The Web

Gnutella is a real-time search, peer-based file-sharing client that allows user running Gnutella to search for and download files from other Gnutella users. Gnutella was originally conceived, authored and released by Justin Frankel and Tom Pepper of [Nullsoft](#) in March 2000.

Net film firm taps Gnutella for video sales

Net film firm taps Gnutella for video sales By Gwendolyn Mariano Staff Writer, CNET News.com June 14, 2000, 5:35 p.m. PT An online movie distributor this week said it will try to sell videos using Gnutella's file-sharing software, issuing a challenge not only for itself, but for the company that has promised to protect files from piracy on the notoriously insecure network: Microsoft. More can be found [Here](#) from Cnet's News.com

Need a host?

Remove

Add

Host: gnet3.ath.cx

Also check out:

gnet1.ath.cx

fileforum.com

This address will send you to one of the host caches, which may not be 100% reliable. This cache listens on ports 23, 80, 6346, 6667, and 8080. The Internet is an unpredictable place, but we try hard to make it work the way it does in movies. If it doesn't work then try again later, or try gnet2.ath.cx! DO NOT E-MAIL US TELLING US IT DOESN'T WORK! Chances are, we already know. Thanks.

[Spread Gnutella!](#)

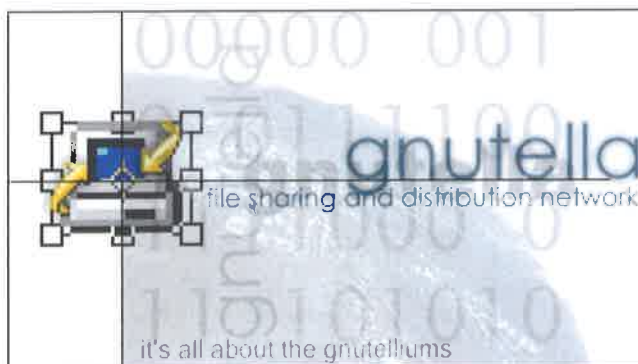


EXHIBIT A

Wego.com, Inc., does not claim ownership of or take responsibility for any content on this site.
There is no warranty or guarantee on anything downloaded from this site.
All trademarks are the property of their respective owners.

[Gnutella Website Support](#)

EXHIBIT A


[Home](#)
[View What's New Pages](#)
[INVITE OTHERS](#) | [HELP](#) | [FEEDBACK](#)
 GUEST: [LOG IN](#) | [JOIN GROUP](#)
[What's New](#)
[Developer's Corner](#)
[Help/Support](#)
[Gnutella?](#)
[Tutorial](#)
[FAQ](#)
[Software](#)
[Third-Party](#)
[Download](#)
[Press](#)
[Promote Gnutella](#)
[About](#)
[Site Map](#)
[Community](#)
[Discussion Board](#)
[Resources](#)
[File Center](#)
[Links](#)
1743392

Since April 10th, 2000

WHAT'S NEW

What's new with Gnutella

[What's New](#)
[Back to all pages](#)
June 11, 2000

Nathan - Changed the picture up top on the webpage along with the color of the webpage. Also added a few more pictures to the [Promote Gnutella](#) section on the page.

June 7, 2000

Nathan - Added a new client called Gnewtella to the [Third-Party](#) section of the website. I've gotten tons of pictures for the Promote Gnutella section of the website. I get like 2 a day. I'm thinking as what I should do now, cause It will take a while to get all the pictures up.

June 1, 2000

Cody - Hello All, not a whole lot going on in the world of Gnutella right now. We are working hard on the next release of our own little client for Unix Gnubile. Looks for a new UI and many bug fix's soon.

June 1, 2000

Nathan - I'm in Seattle, Washington now. I haven't had time to make the much needed changes to the website. There has been a Macintosh gnutella clone added to the third-party section. This is the first Macintosh client/clone that does not require the Java runtime files I believe. In the coming weeks I hope to get the site to the point where we don't get anymore annoying e-mails of people asking us, "How do I download?", which means revisions of the Tutorial. If you have information that you think needs to be added please [e-mail](#) me.

May 25, 2000

We hit the 1,000,000 mark for unique hits on our website. Took us 45 days or 1 & 1/2 months to achieve this mark.

May 24, 2000

Nathan - Added a new client/clone to the [Third-Party](#) area, which is our first BeOS clone.

May 23, 2000

Nathan - Changed some of the wording on the side bar to make it more specific as to what everything was. Changed the [FAQ](#) page so that you can find answers to your questions on certain topics more easily. Changed some of the font and layout of some of the dev pages to match with the rest of the site. Added two new clients to the [Third-Party](#) area. Updated the [Download](#) page so it has those cool 3d tables and made it so it is easier to find which version YOU should download. Also, added two new images to the [Promote Gnutella](#) section of the website!

May 21, 2000

EXHIBIT A

Gnutella was featured on [ABC World News Weekend](#) today. Congratulations to all the developers and users who have taken Gnutella this far!

May 19, 2000

Nathan - Changed the main graphic for the top of the page. E-mail [me](#) and tell me what you think of the new graphic. Also switched the graphical buttons on the side back to their original yellow color. The buttons got messed up during a server move or something or rather. Also changed the color of everything to a little funkier blue.

May 18, 2000

Added a graphic to the [Promote Gnutella](#) section.

May 17, 2000

Nothing major to report, really. Gnutella is growing every day, and the network just gets better and more usable. Client software is improving, and we get client software submissions almost daily. Thanks to all of you for sticking through the thick and thin.

Older stuff

Got this from Kurt Nimmo. Totally funny [Flash animation about Metallica vs. Napster](#).

Got this from [Slashdot](#). Looks like the FTC has found that you and I have overpaid \$480 million for CD's.

Not much here yet, but we'll fill this page with the new day-to-day.

Just added: [mailing lists for all Gnutella client software](#). Tell the authors what you think of their software. Contribute to its development. And learn from others.

[FRIEDCAT](#) has released 1.1b of his great-looking software [N-Tella](#). Sources close to FRIEDCAT tell us he is working hard on getting the bugs out of N-Tella, so we should soon see a functional and beautiful piece of software. We've got 1.1b [mirrored here](#).

And, we're keeping Gnutella in the news! Check out the [press room](#).

Wego.com, Inc., does not claim ownership of or take responsibility for any content on this site.
There is no warranty or guarantee on anything downloaded from this site.
All trademarks are the property of their respective owners.

[Gnutella Website Support](#)


[Home](#)
[View Support Pages](#)
[INVITE OTHERS](#) | [HELP](#) | [FEEDBACK](#)
 GUEST: [LOG IN](#) | [JOIN GROUP](#)
[What's New](#)
[Developer's Corner](#)
[Help/Support](#)
[Gnutella?](#)
[Tutorial](#)
[FAQ](#)
[Software](#)
[Third-Party](#)
[Download](#)
[Press](#)
[Promote Gnutella](#)
[About](#)
[Site Map](#)
[Community](#)
[Discussion Board](#)
[Resources](#)
[File Center](#)
[Links](#)
1743392

Since April 10th, 2000

[Back to all pages](#)

SUPPORT

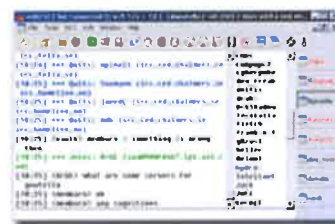
[Main](#) | [FAQ](#) | [Tutorial](#) | [Corporate](#) | [Firewalls](#)

Support : Index

The best place to get help on Gnutella is on the EFNet IRC network, in the #gnutella channel. That's where everybody hangs out and talks about Gnutella. You can find just about everybody that works on this page there. If you don't have IRC, go to irchelp.org, where you can find FAQs, clients, and tutorials on IRC

If you're looking for a Gnutella tutorial or FAQ, check out the links above.

Please do not e-mail us with technical support requests for Gnutella, as you will not get a response. Instead, please use the resources provided on this site.



Support : What Is Gnutella

Gnutella is a fully-distributed information-sharing technology. Loosely translated, it is what puts the power of information-sharing back into your hands.

When the World Wide Web started, that's how it was. It used to be that I would put up a web page, you would link to it, and I would link to yours. To get around, we would all "surf the links". The web was a web. But shortly after, the likes of Yahoo! and Lycos came on the scene to build search engines, or information portals. You go to one place to find all the information. Ideally that would be true. The problem with portals? They stuff you with ads. They are outdated. They basically control the flow of information. (Proven in a recent IBM/Altavista study of the Internet.)

Now, however, Gnutella puts the personal interaction back into the Internet. When you run a Gnutella software and connect to the Gnutella Network, you bring with you the information you wanted to make public. That could be nothing, it could be one file, a directory, or your entire hard drive (I wouldn't recommend this option).

The power behind this is amazing. That data which you have bothered to keep on your hard disk is what you found to be valuable. So when you share it you are sharing what is most valuable on the entire Internet. And you control its sharing. Decide to stop sharing? Go ahead and take those files offline. Want to share more? Select more files and share them. It's really that easy, and the power of sharing information is just unquantifiable.

Gnutella client software is basically a mini search engine and file serving system in one. When you search for something on the Gnutella Network, that search is transmitted to everyone in your Gnutella Network "horizon". If anyone had anything matching your search, he'll tell you.

So, time to give a brief explanation of the "horizon". When you log onto the Gnutella network, you are sort of wading into a sea of people. People as far as the eye can see. And further, but they disappear over the horizon. So that's the analogy. When you log on, you see the host counter start going crazy. That's because everyone in your horizon is saying "Hello" to you. After a while, it stops counting so rapidly, because you've counted most everyone in your horizon. Over time the people in the horizon change, so you'll see the

counter move slowly.

If you log in another day, you should see a whole bunch of fresh faces, and maybe you'll have waded into a different part of the network. A different part of the crowd. Different information.

You might say, "Well, what if what I want doesn't exist within my horizon? The horizon never seems to grow beyond 10000 hosts!" True. If the information you want isn't in your horizon, you're out of luck. But hopefully it is. It's a probability game. Fortunately information is duplicated. Everyone's got that joke about the way men choose urinals, for example.

And what of the 10000-user horizon? That's just the network "scaling". The Gnutella Network scales through segmentation. Through this horizoning thing. It wouldn't do to have a million people in the horizon. The network would slow to a crawl. But through evolution, the network sort of organizes itself into little 10000-computer segments. These segments disjoin and rejoin over time. I leave my host on overnight and it will see upwards of 40000 other hosts.

Gnutella Is Filesharing

Okay, so you've got Napster, and you've got SpinFrenzy, and you've got CuteMX. And you've got FTP sites, web pages, and all kinds of other stuff. And you still can't find what you want.

Enter Gnutella. Gnutella is the answer to your prayers. Searching for that recipe for strawberry-rhubarb pie? Can't find the latest version of Linux on a T3 or better link? It's probably out there on the GnutellaNet.

What's great about Gnutella is that it isn't focused on trading MPEG music the way the other guys are. There is all kinds of stuff on the GnutellaNet, and when you get a search hit, it's virtually guaranteed to be there. No stale links. No irrelevant hits. And if you don't want to download your recipes from someone in Belize with a 2400 bps link, you just set that in your query.

The other half of Gnutella is giving back. Almost everyone on GnutellaNet *shares* their stuff. Every client on the GnutellaNet is also a server, so you not only can find stuff, but you can also make things available for the benefit of others. So if you've got a good recipe for blueberry cobbler, you could answer someone's prayers by sharing it with the rest of the GnutellaNet!

Gnutella Is Anonymous

One of the problems with Napster and others like it is that they are centralized. A centralized place for government agencies to impinge upon your freedom to search the net. All those commercial realtime search engines probably keep logs so they can target ads at you. At least they keep logs so they know how many searches they get per day so they can tell it to their investors. And they probably run some data mining to figure out how many people searched for MP3's, how many people searched for recipes, etc. All that so they can figure out *exactly* what their customer is like.

Gnutella puts a stop to all those shenanigans. When you send a query to the GnutellaNet, there is not much in it that can link that query to you. I'm not saying it's totally impossible to figure out who's searching for what, but it's pretty unlikely, and each time your query is passed, the possibility of discovering who originated that query is reduced exponentially. More on that in the next section.

In short, there is no safer way to search without being watched.

A big however, however. To speed things up, downloads are not anonymous. Well, we have to make compromises. But again, nobody's keeping logs, and nobody's trying to profile you.

Gnutella Is The Game : Telephone

EXHIBIT A

Remember the game telephone? Yeah, you know...someone whispers something to you, and you're supposed to whisper that to someone else, and so on and so on? And the guy at the end of the chain got a whisper that sounded nothing like the original whisper. Well, Gnutella is similar in many ways.

When you say to GnutellaNet, "Hey, find strawberry-rhubarb pie recipes.", you are actually saying, "Hey, my close friends, could you tell me if you've seen any recipes for strawberry-rhubarb pie? And while you're at it, ask your close friends too. And ask them to ask their friends." It's obvious that after just a few rounds of this, you've got a lot of friends working on finding that recipe! And, it's pretty much impossible for any one person to know who asked the question in the first place. Two beautiful things about Gnutella.

So suppose some guy, 6 degrees from you (your friend's friend's friend's friend's friend's friend), has the world's best recipe for strawberry-rhubarb pies. He tells the guy who asked him. That guy tells the guy who asked him... And ultimately the answer gets back to you. But only one person in the whole world knows that you're the original person who asked. And guess what? In GnutellaNet, we even fix that. The guy you asked originally doesn't even know that you're the person who's really asking the question. Err...searching for that strawberry-rhubarb pie recipe. So you get your answer, and you don't have to admit to anyone that you're a pantywaist.

Could it get any better?

Gnutella Is Designed to Survive Nuclear War

Gnutella is designed to survive nuclear war. It's true. Maybe it's a munition. Who knows. What we do know is that GnutellaNet cannot be defeated by something so lame and simple as an ICMP flood. It's not going to leave you stranded like Yahoo! did when it was pummeled by Smurfs.

The GnutellaNet concept is not to have one massive Gnutella service provider that can go down and bring the whole world to a screeching halt. If, say, New York was hit by Dr. Evil, GnutellaNet probably wouldn't even notice. So you lose your "Gnutella friends" in New York. Big deal. It's unlikely that New York is the only place where people share the recipe for strawberry-rhubarb pie.

You don't have to stand around in your kitchen for the next three hours waiting for some geek to switch the DNS from GlobalCenter to Exodus. You just issue your query, and some guy in Texas jumps up and down and waves his Gnutella hands in the air and says, "Pardner, I've got a recipe for strawberry-rhubarb pie that'll have you salivating like a dog in heat."

Gnutella Can Withstand A Band of Hungry Lawyers

Gnutella can withstand a band of hungry lawyers. How many realtime search technologies can claim that? Not Napster, that's for sure. Just to emphasize how revolutionary this is: hungry lawyers are probably more destructive than nuclear weapons.

There are a few things that will prevent Gnutella from being stopped by lawyers, FBI, etc. First, Gnutella is nothing but a protocol. It's just freely-accessible information. There is no company to sue. No one entity is really responsible for Gnutella.

Second, Gnutella is not there to promote the piracy of music. It's a *technology*, not a music-piracy tool.

The important thing is that Gnutella will be here tomorrow. It's reliable, it's sharing terabytes of data, and it is absolutely unstoppable.



INVITE OTHERS | HELP | FEEDBACK
GUEST : [LOG IN](#) | [JOIN GROUP](#)

Home

View Support Pages

What's New
Developer's Corner
Help/Support
Gnutella?
Tutorial
FAQ
Software
Third-Party
Download
Press
Promote Gnutella
About
Site Map

Community

Discussion Board

Resources

File Center
Links

1743392

Since April 10th, 2000

[Back to all pages](#)

SUPPORT

[Main](#) | [FAQ](#) | [Tutorial](#) | [Corporate](#) | [Firewalls](#)

Support : [FAQ](#)

Frequently Asked Questions

Maintained By [manuka](#)
Last updated: May 28, 2000

Source Code

[What Is Gnutella Written With?](#)
[Will The Source Code Be Released, and if so, when?](#)

[Where Does The Name Gnutella Come From?](#)

Protocol

[How did you get the protocol specs?](#)

Availability

[Is it available for Linux?](#)
[Is it available for Mac?](#)

Clients

[What's the best client to use?](#)

Downloading

[Do the file transfers go through the GnutellaNet?](#)
[How do I know what IP I'm downloading from?](#)
[Does Gnutella automatically resume interrupted downloads?](#)
[Does anyone have <file> for me to download?](#)
[Why are my downloads slow?](#)
[How come when I try to download songs from Gnutella nothing happens?](#)
[If people are downloading from me will this slow down anything else I'm doing on the Internet?](#)

Searching

[Why Does A Search For *.mp3 Not Work Correctly?](#)
[Does a search normally take a long time?](#)
[Why doesn't my search for <file> come up with anything?](#)
[Is there anyway to stop or cancel a search?](#)
[Since my IP address isn't sent along with my search, how does other people's Gnutella clients find out where to send the search results?](#)

Firewalls

[I'm behind a firewall. Can I still use Gnutella?](#)

File Sharing/Security

[Doesn't gnutella expose me to virus attacks?](#)
[Should I be worried about the potential hazards of giving anyone access to my](#)

[computer?](#)

[Will people only have access to my files when Gnutella is running or every time I am on the Internet?](#)

Servers/Hosts

[I'd like to run a Gnutella server. Who do I ask?](#)

[Where Can I Find A Good Server To Connect to?](#)

[How many hosts should I connect to?](#)

IO/Stats

[What's the difference between incoming and outgoing connections?](#)

[What does the "Speed" number mean?](#)

[What are dropped packets and how do I reduce the number of dropped packets I receive?](#)

[What are those three comma-separated numbers in the info field after each connection \(S,R,D\)?](#)

[Where Are Settings Saved?](#)

Copyright/AOL/RIAA/Misc

[What's this "The RIAA is Watching You" message about?](#)

[Is this program designed for exchanging copyrighted files?](#)

[I thought AOL pulled the plug at 0.48, so what's with these new versions? Who is producing them? Are they official? Are more coming?](#)

[I'm a parent concerned about pornography. Is this a haven for perverts?](#)

[Gnutella Sucks! You're all a bunch of lamers!](#)

What's this "The RIAA Is Watching You" message about?

Many people report seeing this message when performing a search. This is clearly the result of some moron who has hacked up one of the clients to return that in response to every search. Don't worry about it, they're just abusing the network. Ditto for any other similar messages.

Where Can I Find A Good Server To Connect to?

Gnutella's peer-to-peer model uses no servers. Each person is an individual node in a network of computers. If you need a server please use the reflectors listed on the [main page](#) or in the topic of the IRC channel. If those don't work, try logging on to IRC and getting someone's IP address via your client's lookup command (usually /dns) – consult your client's documentation on how to do this if you're unsure. Please do not ask the entire channel for somebody that has an IP for Gnutella, as you will likely find yourself out of the channel very rapidly. Many people in #gnutella are running Gnutella or one of the third-party clients. The default port is 6346. Keep trying until you get one. Alternately, try downloading the gnutella.net file from the [File Center](#) and putting it in your gnutella directory, overwriting the existing one. This file is updated fairly regularly. You can also check out my Gnut [hosts list](#), which is relatively current.

What's the difference between incoming and outgoing connections?

The fact that people ask this one mystifies me. The answer is very simple. One is incoming connections, the other is outgoing connections. For those of you still unclear on this concept, an incoming connection is where someone is connected to you. And outgoing one is where you're connected to someone else. Does matter what direction it is, data is passed both ways.

Will The Source Code Be Released, and if so, when?

If all goes well and development of Gnutella continues you should see the source code during the 1.0 release. The source code has not been released yet and we are unsure at this time if it will ever be released, due to AOL locking it in a closet underneath a large pile of Time Warner's dirty laundry.

What does the "Speed" number mean?

Oddly enough, this is asked numerous times a day on IRC. The speed setting reflects, oddly enough, the speed of the line on the remote end that returned the search. This is not automatically determined, and is set in the configuration on the client, and as such is rarely accurate.

Where Are Settings Saved?

Settings are saved when you shut down Gnutella and are saved in the GNUTELLA.INI & GNUTELLA.NET files in your Gnutella directory.

Where Does The Name Gnutella Come From?

The GNU in Gnutella comes from GNU, which is obvious. The other part, "Nutella", refers to a chocolate and hazelnut spread that is phenomenally popular among Europeans.

<http://www.ferrero.com.ar/productos/nutella/nutella.html>

What Is Gnutella Written With?

Microsoft Visual Studio

Is it available for Linux?

There are third-party Linux clients already available! Some of the Beta versions of Gnutella for Windows are reported to work with WINE.

Is it available for Mac?

There is a Java version that is reported to work nicely under MacOS.

Do the file transfers go through the GnutellaNet?

No. The GnutellaNet is used only for exchanging servant messages and performing searches. The actual transfer is done directly between servants via HTTP.

How did you get the protocol specs?

Some smart people analyzed the traffic, and reverse engineered most of it. Other parts were obtained from a conversation on IRC.

I thought AOL pulled the plug at 0.48, so what's with these new versions? Who is producing them? Are they official? Are more coming?

In the interest of protecting those involved, it can only be said that these versions are being produced by someone with access to the source code. No, of course they're not *official* in the GnuSoft sense, but they do come from a trusted source. I would certainly expect more versions, since we've already seen several materialize.

What are those three comma-separated numbers in the info field after each connection? (S,R,D)

Those (at present) represent the number of GnutellaNet messages sent, received, and dropped, to that host.

How do I know what IP I'm downloading from?

There is a column in the downloads window immediately to the right of the "Status" column. It is set to zero width by default, but it can be stretched.

Why Does A Search For *.mp3 Not Work Correctly?

The * is ignored. if you really, really want to search for all mp3's out there (we think you're nuts if you do, however), search for ".mp3". Or whatever extension your little heart desires.

Is this program designed for exchanging copyrighted files?

Gnutella is a tool for general peer-to-peer file sharing. It can be used to share spreadsheets, source code, design documents, really any file on your computer. Yes, it is possible to exchange illegal files. This is entirely the choice of the people sharing them. That is, after all, the beauty of freedom of choice. We do not condone or endorse the exchange and transfer of such files, and would like to point out that doing so is entirely at your own risk. The files you can find out on the GnutellaNet include recipes from Usenet's rec.food.recipes group, as well as files from the [Gutenberg Project](#).

I'd like to run a Gnutella server. Who do I ask?

There are no servers. This is a decentralized system that does not rely on central servers that can be shut down on a whim. If you want to be part of the network, download the Gnutella application, and fire away.

I'm behind a firewall. Can I still use Gnutella?

Certainly. If you're exchanging files with people outside your firewall, the other end of the transaction can't be firewalled. But you can also set up your own GnutellaNet within your firewall, simply by having others connect to each other. Please see the [firewalling document](#) for details on Gnutella and firewalls.

How come when I try to download songs from Gnutella nothing happens?

Usually if nothing happens, it's because the person who is hosting the files has recently renamed the files they are sharing and has forgotten to rescan the folders in which they are sharing the files, or is behind a firewall.

I'm a parent concerned about pornography. Is this a haven for perverts?

Given that gnutella is designed to share any type of files, it's quite likely that there are people sharing adult material. There is really nothing that can be done to stop this. However, the proportion of searches for adult material on gnutella is not significantly different from what you will find on the web at large. There are several people on the network actively trying to reduce the amount of "objectionable" material flowing across their systems.

Doesn't gnutella expose me to virus attacks?

No more so than downloading files from the Internet through any other means such as FTP, the Web, IRC, or Usenet. Use common sense and scan executable file using a current virus scanner with a recent list.

What are dropped packets?

A packet is "dropped" when the client at the other end of the connection is unable to keep up with the rate at which the data wants to flow. It's trying to drink from a fire hose. If you find that a particular client connection has a lot of dropped packets and that number doesn't stop growing, then you should close that client's connection because it's wasting

EXHIBIT A

your time and dragging the entire network down. That system isn't receiving much of the data it is supposed to, so it really doesn't need to be connected to you. Aggressively closing connections to clients which are dropping an unreasonably high number of packets benefits the performance of the network at large. To reduce the number of dropped packets that you receive check all the hosts you are connected to for dropped packets and remove any where the number is increasing rapidly.

Does Gnutella automatically resume interrupted downloads?

Whenever you close Gnutella and you have a file that was still downloading, it will resume where it left off upon restarting Gnutella. This doesn't always work, as the system you were downloading from may no longer be on the network.

Does a search normally take a long time?

The amount of time it takes for a search to complete depends on your speed and the speed of the others you are connected to. It takes time to traverse the GnutellaNet made up of many high and low speed connections. Give the search at least a few minutes. **Why doesn't my search for <file> come up with anything?**

The most likely cause is that you searched for something nobody has shared out at the moment. But just to be sure, make sure you have at least one connection to the GnutellaNet, and that you can see more than one host.

Since my IP address isn't sent along with my search, how do other Gnutella clients find out where to send the search results?

Gnutella keeps track of all the IP addresses on the network along with each client's TTL and hops. Gnutella then matches up the TTL and hops from the search with the ones it has from all of its pings and sends it to the IP address it found.

Is there anyway to stop or cancel a search?

Since there is no point that can define the "end" of a search, it will keep accepting results until you start another.

What's the best client to use?

I have no idea what your needs are, and neither does anyone else. The best client is whatever works for you and meets your individual needs. Why don't you try one?

Does anyone have <file> for me to download?

Gnutella is a file search tool, so why don't you go search for what you're looking for with Gnutella? We're not your personal locator service, but if you still insist on having us find stuff for you, I'm sure we can accommodate you for a reasonable fee, just provide us with your billing information and credit card number, and we'll be glad to help.

How many hosts should I connect to?

If you're using a dialup connection, no more than two. Bear in mind that each connection uses up between 500 and 1000 bytes/second of bandwidth. If you're on a faster connection, four is a good number.

Why are my downloads so slow With Gnutella?

The speed of your file transfers has nothing to do with Gnutella itself, but rather the network connection between the two systems. The actual file download is a direct HTTP connection and would be no faster than if you used a web browser to the same system. Bear in mind you're quite likely not the only person downloading files from the system on the other end. If they're on a dialup, transfers will be slow.

Should I be worried about the potential hazards of giving anyone access to my computer?

The only files people can see are the ones you allow them to see. If you have important information in any of the folders that you are sharing then it would be best to move them to another folder. Gnutella only shares the folders which you specify - this can be changed in the configuration area of Gnutella.

Gnutella Sucks! You're all a bunch of lamers!

Nobody said you had to use Gnutella. Use whatever you like. We really don't give a damn about your holy war and aren't going to play your silly troll games.

Will people only have access to my files when Gnutella is running or every time I am on the Internet?

Whenever you have Gnutella open and you are connected to the internet and connected to some body on Gnutella then the people who you are connected have access to the files you are sharing on the network. You have to be connected to the internet and to a Gnutella host in order to share files with your neighbors.

If people are downloading from me will this slow down anything else I'm doing on the Internet?

Whether or not you have a fast enough connection will determine if anything else is slowed down because of Gnutella. In fact, Gnutella does take up quite a lot of bandwidth and if you have a slow connection the most likely your other connections to the internet (whether it be via WWW,FTP, ect..) will suffer.

Wego.com, Inc., does not claim ownership of or take responsibility for any content on this site. There is no warranty or guarantee on anything downloaded from this site. All trademarks are the property of their respective owners.

[Gnutella Website Support](#)



INVITE OTHERS | HELP | FEEDBACK
GUEST : [LOG IN](#) | [JOIN GROUP](#)

[Home](#)

View Support Pages

What's New

Developer's Corner

Help/Support

Gnutella?

Tutorial

FAQ

Software

Third-Party

Download

Press

Promote Gnutella

About

Site Map

Community

Discussion Board

Resources

File Center

Links

1743392

Since April 10th, 2000

[Back to all pages](#)

SUPPORT

[Main](#) | [FAQ](#) | **Tutorial** | [Corporate](#) | [Firewalls](#)

Support : Tutorial

Tutorial

Maintained By [nouser](#)

Last update: May 23, 2000

[What Is Gnutella?](#)

Please read the section of the site about what exactly Gnutella is.

[How To Download](#)

[How To Install](#)

[How To Uninstall](#)

[How Gnutella Works](#)

[Let's Get Started](#)

[Interface Rundown](#)

[Getting Connected](#)

[How To Search](#)

[Searching Tips](#)

[Limiting Searches](#)

[Client Configuration](#)

Ok, so maybe you've already downloaded Gnutella and maybe you're not quite sure how to use it. I don't blame you. At first glance, yeah, it might seem like something for the techie but once you learn how to use it'll be a tool you'll use everyday.

How To Download

Click on the Download section after you've gone to <http://gnutella.wego.com/>. Download the latest version, which should be listed first.

- Hit that link and choose to [Save File](#).
- Once you have downloaded it double-click on the file and install it using the install program, which should take you step by step through the installation process. See: [How To Install](#).

How To Install

If you are prompted with the install program leave the type of install as normal and hit next. At this point you will be asked to choose a directory to install the program in. The default is "c:\Program Files\gnutella". This screen will also tell you if you have enough room to install Gnutella, after you have specified a directory hit next and the files will be installed. After you have installed Gnutella you will have shortcuts on the desktop, and the start menu. Hit one of these shortcuts and Gnutella will be launched.

How To Uninstall

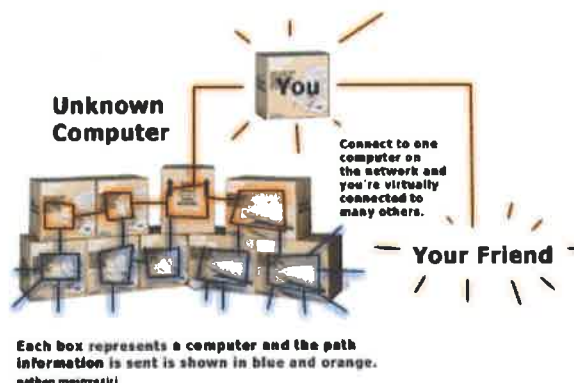
There is no uninstall program for Gnutella, which means that you'll have to delete Gnutella manually. Gnutella saves none of its settings in the registry so there should be no

EXHIBIT A

problem there. All you need to do is delete the folder where you installed Gnutella. For most people they would goto My Computer, then to Program Files and delete the Gnutella folder they see. If you didn't install it in the default directory, you're not going to find it here.

How Gnutella Works

Before we start, we need to get some things clear. For one, Gnutella doesn't use servers to connect to like Napster or CuteMX. You connect to one person on the network, by entering their IP in the IP text box and that connects you directly to that one person on the network. When you search, though, it will do a recursive search throughout the network starting with that one person you are connected to.



As you can see the box that says, "You" connects to another box or computer. "You" is directly connected to that box and indirectly connected to all of the other boxes. When a search is done, "You" send the search to the box it is directly connected to, and that box sends the search to all the boxes it is connected and so on and so forth.

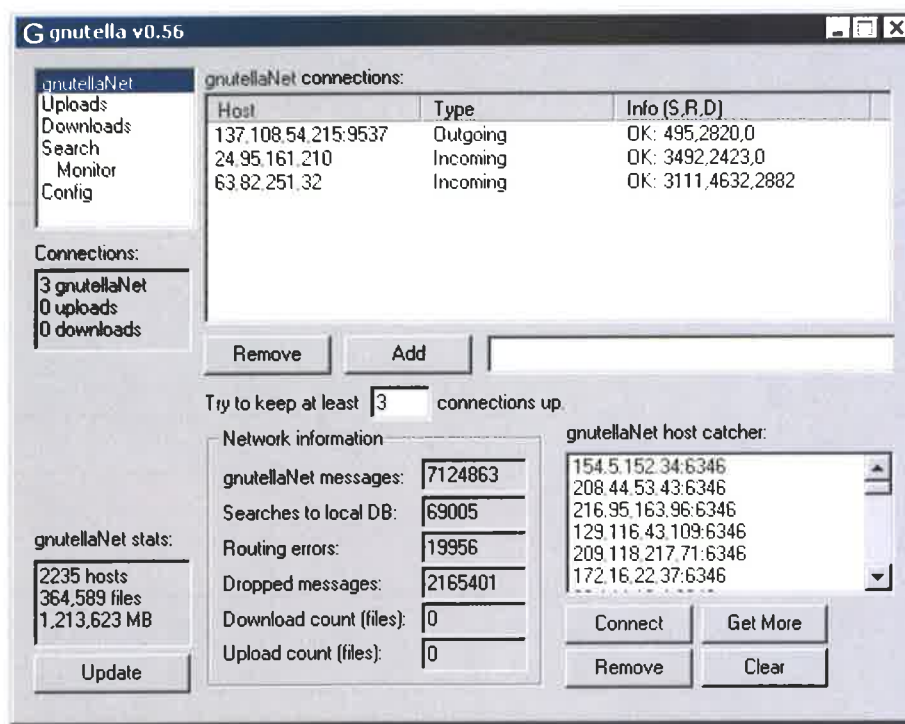
For Techies

Gnutella is a peer-to-per network. Each person on the network is considered a hub.

In other terms, everybody on the network is a client and a server.

Let's Get Started

[Back to The Top](#)

EXHIBIT A**Just Tell Me How To Get Connected!**

When you first load up Gnutella, it doesn't automatically connect to any servers or anything which is why you have to find somebody to connect to before you can start search for files and downloading them.

First thing you need to do is find one of the computers on the network. You need to at least have the [IP](#) of the computer that is on the network.

You would enter IP address in the format as shown below:

IP : Port

findshit.gnutella.com:8015 and 24.15.16.29 are examples.

If you are connecting to somebody on the default port (6346) then you don't need to add ':6346'

The IP addresses listed in this tutorial page probably won't work for you. [To get an IP](#) on the network you might want to check out this page. You can download the gnutella.net file listed there and put it in your Gnutella directory. IP addresses are also often up at the main gnutella page, <http://gnutella.wego.com/> Don't give up on the first IP address that you try, it may take several tries before you actually get connected to any part of the network. If you can't seem to find any IP addresses there you might want to check out the channel topic on #gnutella on the EFnet IRC network (use [mIRC](#) if you don't have a windows IRC client).

Interface Rundown

When you first look at Gnutella for Windows it may appear not appear to be the powerful program that it is, but Gnutella covers a good range of functionality. The upper left-hand corner contains the tabs GnutellaNet, Uploads, Download, Search, Monitor, and Config.

Feature

Function

EXHIBIT A

GnutellaNet	The main screen It shows your connected hosts, network information, host catcher, and a text box to add servers
Uploads	Shows what people are downloading from you and their download progress
Downloads	Shows you are downloading and your download progress
Search	Where you search for files to download. You must be connected to somebody in order to download or even search for files to download
Search Monitor	In this screen you can see what people are searching for. <u>NOTE: THE SEARCH MONITOR IS NOT FOR CHATTING.</u>
Config	Where you can change the options relating to the way that your Gnutella client functions

Techie Explanation Of The Interface

Next lets start by going over the picture above and what it means to you.

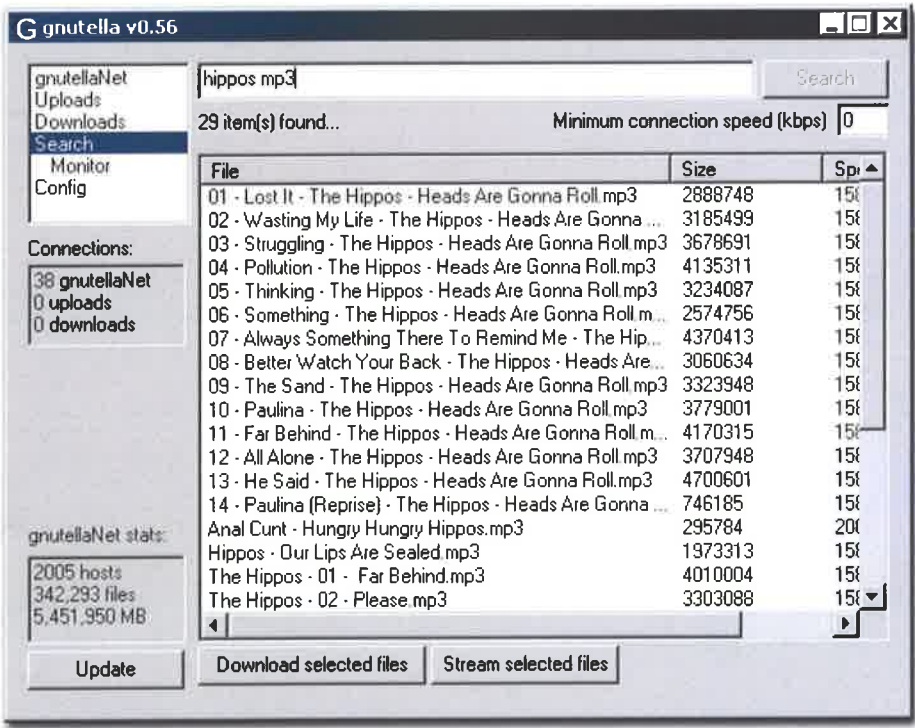
List	Definition of Contents
Connection List	This is the list of people you are directly connected to
Header	Definition of Contents Under The Header
<u>Host</u>	IP of computer you are connected to
<u>Type</u>	Type of connection. If you add an IP to your connection list the type is considered outgoing and the other way around is an incoming connection.
<u>Info</u>	Packets Sent , Packets Received , Lost/ Dropped Packets
List	Definition of Contents
Connection Stats	This is the list of people you are directly connected to
Header	Definition of Contents Under The Header
<u>gnutellaNet</u>	Number of direct connections you have open
<u>Uploads</u>	Number of uploads in progress
<u>Downloads</u>	Number of downloads in progress
List	Definition of Contents
GnutellaNet Stats	Contains information on network you and your connections make
Header	Definition of Contents Under The Header
<u>Hosts</u>	Number of direct connections and the number of the direct connections' connections. This makes up your network.
<u>Files</u>	Number of files in the network
<u>MB</u>	Size of files in the network

EXHIBIT A

<u>Ms Avg. Ping</u>	Average ping time in milliseconds to all the Hosts in the network
Feature	Function
Add/Remove Button	Adds or removes an IP from your connections list.
Keep # Connections Up	How many connections to other computers Gnutella should try to keep open. If this number is lower than the number of connections Gnutella has open then Gnutella adds hosts from the <i>Host Catcher</i> list and tries to connect to them.
Host Catcher	When your stats are updated or information is sent from one computer to another in which an IP is required to send the data, the IP is added to this list. When Gnutella is then in need of hosts to connect to it uses the IP addresses in this list.

How To Search

[Back to The Top](#)



Again, you have to be connected to somebody for searching to work. (See [Just Get Me Connected!](#))

To search for anything on the network just enter what you are search for in the search text box and press the 'Search' button.

The search button works like the button on an elevator. No matter how many times you search for the same thing over and over, it will not make the searching go any faster than it is.

If you are not getting as many search results as you would like, try connecting to another computer on the network which will allow you to connect to even more computers on the network. Dialup users shouldn't connect to more than 1 or 2 other peers, and

high-bandwidth users should limit themselves to 10 connections. Too many connections will consume all your available bandwidth (each one uses 500-1000 bytes/second) and cause dropped packets due to redundancy on the network. Also make sure you have more than 1 host in the Gnutella stats on the side screen. You need to press 'update' to manually check how many people you are direct and indirectly connected to.

Searching Tips

Most of the time if you want to search for file types other than mp3's then you'll need to specify the file type as shown above.

- Periods and wild cards and Regular Expressions are not supported
- Search terms also look in a file's pathname
- Errors on uploads/downloads either because of firewall/proxy or routing to other system
- *.AVI is the same as AVI
- Search terms are separated by spaces and wild cards and periods are converted to spaces on the remote computer when searching

When you do a search using Gnutella, it searches the files of everybody you have in your list and searches each one of their lists as well for what you're looking for. This is what you call the hosts.

Limiting Searches

There are a few ways to limit searches. One way is by using the Minimum Connection Speed (kbps) as shown in the graphic above. The Minimum Connection Speed will only search people who are higher than the speed that you've specified.

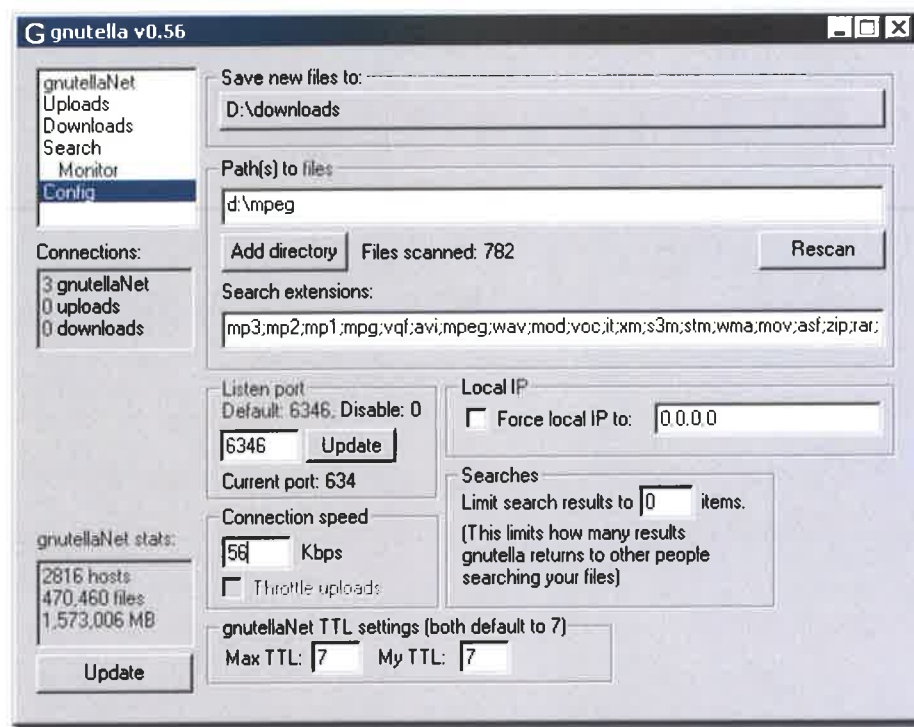
Techie Explanation Of The Interface

Feature	Function
Stream Selected Files	This only works for MP3 files and you must have WinAMP to begin with. All this does is put the MP3 file name in your WinAMP playlist so that you can list to the song as it downloads.
Download Selected Files	Downloads the selected files. You can see the progress of all downloads in the Download screen

Client Configuration

[Back to The Top](#)

EXHIBIT A



Feature

Function

Save New Files To

AKA *Your Download Directory*. Specifies the directory in which all files that are downloaded will be placed. Click on the button to change the directory.

Add Directory Button

Adds a directory to the list of shared folders. You have to press 'Rescan' before files in any new directories are shared.

Rescan Button

Sets the Files Scanned property at 0 and starts to scanning each one of the shared folders for files of all types. When it is done Files Scanned will show how many files you are sharing in the directories scanned.

Path(s) To Files

AKA *Your Shared Folders* Specifies the folders that are scanned and then shared to the outside world! Each directory path is separated by a semi-colon.

Search Extensions

When you do search for something, only files with the file types specified here, will be shown. Notice that each file type does not include a period at the beginning, and that each one is separated by a semi-colon.

Searches (Limit search results to...)

This does not limit the number of results you get when you search for something, but only limits the number of results Gnutella returns when other people are searching through your files.

Word

Definition

Listen Port

Port in which people connect to you.

Time To Live

Number of computers a single packet goes through before it is stopped

Connection Speed

Your connection speed in KBPS

Files Scanned

Number of files you are sharing

Wego.com, Inc., does not claim ownership of or take responsibility for any content on this site.
There is no warranty or guarantee on anything downloaded from this site.
All trademarks are the property of their respective owners.

[Gnutella Website Support](#)

EXHIBIT A


[Home](#)

View Client Software Pages

[INVITE OTHERS](#) | [HELP](#) | [FEEDBACK](#)
 GUEST : [LOG IN](#) | [JOIN GROUP](#)
[Back to all pages](#)

What's New

Developer's Corner

Help/Support

Gnutella?**Tutorial****FAQ**

Software

Third-Party

Download

Press

Promote Gnutella

About

Site Map

Community

Discussion Board

Resources

File Center

Links

1743392

Since April 10th, 2000


CLIENT SOFTWARE
[Index](#) | [Gnutella](#) | [Third-party](#) | [Submit Your Client](#)

Client Software : Gnutella

Official Releases				
Name	D/L	Version	Comments	
Gnutella		0.56	97KB	TTL issue resolved
Gnutella		0.50c	101KB	100% CPU Usage At Times Memory Leak? Download Problems Incorrect Search Result # Found?
Gnutella		0.49	100KB	GPF Errors When Certain Packets Sent
Gnutella		0.45	100KB	
Specially Modified				
Name	D/L	Version	Comments	
Gnutella Brokkolo		0.56	93KB	Modified by brokkolo@operamail.com for improved aesthetics.
Gnutella BR2		0.56	53KB	In Brazilian-Portuguese. Modified to have a bigger window. Updated 18 May 2000.
Gnutella France		0.56	128KB	Now in French. Take a look at the website , also in French.
Gnutella Wieman		0.56	104KB	Modified to have a bigger window. ONLY WORKS CORRECTLY AT 1024x768 RESOLUTION.

Wego.com, Inc., does not claim ownership of or take responsibility for any content on this site.
 There is no warranty or guarantee on anything downloaded from this site.
 All trademarks are the property of their respective owners.

[Gnutella Website Support](#)

DISTRIBUTED SEARCH SERVICES
BY Clip2

Alpha Release

GNUTELLA OPTIMIZED HOST LIST

Need a host? Clip2 DSS brings you the best host list available. To use it, simply add a connection to **gnutellahosts.com:6346** in your Gnutella client. Our server will fill your host catcher with the list.

What makes this list so good? The hosts in it are connected to the Gnutella network in such a way that they can reach more files than other hosts. We refresh the list *continuously* and last confirmed all hosts were responding as of 8/15/00 17:10:07 PDT.

IP ADDRESS	FILES REACHABLE	GB REACHABLE	% OF NETWORK CONTENT
141.84.102.25:6346	294,468	18,748	100.0%
195.24.30.203:6346	294,468	18,748	100.0%
134.102.206.55:6346	294,468	18,748	100.0%
200.248.248.200:6000	294,468	18,748	100.0%
147.52.113.13:6346	294,468	18,748	100.0%
165.230.77.83:6346	294,468	18,748	100.0%
18.244.1.196:6346	294,468	18,748	100.0%
163.176.52.37:6346	294,468	18,748	100.0%
207.235.120.176:5634	294,468	18,748	100.0%
148.241.61.11:5635	294,468	18,748	100.0%

GNUTELLA RESOURCES

- [Gnutella Protocol Specification v0.4 \(PDF\)](#)
- [Example Gnutella network map](#)
- [The "Napster Flood" of July 26-28, 2000](#)
- [What were Gnutella users looking for in June and August, 2000?](#)

GNUTELLA NETWORK STATISTICS

Stats as determined by our crawler:

Last Crawl 8/15/00 16:51:46 PDT

Hosts 1,968

Files* 294,468

Gigabytes* 18,748

[Diameter](#) 9

* As reported by network hosts

GNUTELLA SEARCH

[TellaFind.com](#) uses the Clip2 DSS host list to connect to the Gnutella network.

Search the Gnutella network via TellaFind:

Search

GNUTELLA LINKS

News	gnutellanews.com
Downloads	gnutelliums.com
File sharing portal	zeropaid.com
De facto home	gnutella.wego.com
Newsgroup	alt.gnutella

[Home](#) | [About Us](#) | [Join Us](#) | [Contact Us](#)

© Copyright Clip2.com, Inc. 2000. All rights reserved.

<http://web.archive.org/web/20000818053509/http://gnutellahosts.com/GnutellaProtocol04.pdf>

The Gnutella Protocol Specification v0.4

Clip2 Distributed Search Services
<http://dss.clip2.com>
dss-protocols@clip2.com

Gnutella (pronounced "newtella") is a protocol for distributed search. Although the Gnutella protocol supports a traditional client/centralized server search paradigm, Gnutella's distinction is its peer-to-peer, decentralized model. In this model, every client is a server, and vice versa. These so-called Gnutella *servents* perform tasks normally associated with both clients and servers. They provide client-side interfaces through which users can issue queries and view search results, while at the same time they also accept queries from other servents, check for matches against their local data set, and respond with applicable results. Due to its distributed nature, a network of servents that implements the Gnutella protocol is highly fault-tolerant, as operation of the network will not be interrupted if a subset of servents goes offline.

Protocol Definition

The Gnutella protocol defines the way in which servents communicate over the network. It consists of a set of descriptors used for communicating data between servents and a set of rules governing the inter-servent exchange of descriptors. Currently, the following descriptors are defined:

Descriptor	Description
Ping	Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.
Pong	The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.
Query	The primary mechanism for searching the distributed network. A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.
QueryHit	The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.
Push	A mechanism that allows a firewalled servent to contribute file-based data to the network.

A Gnutella servent connects itself to the network by establishing a connection with another servent currently on the network. The acquisition of another servent's address is not part of the protocol definition and will not be described here (Host cache services are currently the predominant way of automating the acquisition of Gnutella servent addresses).

Once the address of another servent on the network is obtained, a TCP/IP connection to the servent is created, and the following Gnutella connection request string (ASCII encoded) may be sent:

GNUTELLA CONNECT/<protocol version string>\n\n

where <protocol version string> is defined to be the ASCII string "0.4" (or, equivalently, "\x30\x2e\x34") in this version of the specification.

A servent wishing to accept the connection request must respond with

GNUTELLA OK\n\n

Any other response indicates the servent's unwillingness to accept the connection. A servent may reject an incoming connection request for a variety of reasons - a servent's pool of incoming connection slots may be exhausted, or it may not support the same version of the protocol as the requesting servent, for example.

Once a servent has connected successfully to the network, it communicates with other servents by sending and receiving Gnutella protocol descriptors. Each descriptor is preceded by a Descriptor Header with the byte structure given below.

Note 1: All fields in the following structures are in network (i.e. big endian) byte order unless otherwise specified.

Note 2: All IP addresses in the following structures are in IPv4 format. For example, the IPv4 byte array

0xD0	0x11	0x32	0x04
byte 0	byte 1	byte 2	byte 3

represents the dotted address 208.17.50.4.

Descriptor Header

	Message ID	Paybad Descriptor	TTL	Hops	Payload Length		
Byte offset	0	15	16	17	18	19	22

Descriptor ID	<i>A 16-byte string uniquely identifying the descriptor on the network</i>
Payload Descriptor	<i>0x00 = Ping</i> <i>0x01 = Pong</i> <i>0x40 = Push</i> <i>0x80 = Query</i> <i>0x81 = QueryHit</i>
TTL	<i>Time To Live. The number of times the descriptor will be forwarded by Gnutella servants before it is removed from the network. Each servant will decrement the TTL before passing it on to another servant. When the TTL reaches 0, the descriptor will no longer be forwarded.</i>
Hops	<i>The number of times the descriptor has been forwarded. As a descriptor is passed from servant to servant, the TTL and Hops fields of the header must satisfy the following condition:</i> $TTL(0) = TTL(i) + Hops(i)$ <i>Where $TTL(i)$ and $Hops(i)$ are the value of the TTL and Hops fields of the header at the descriptor's i-th hop, for $i \geq 0$.</i>
Payload Length	<i>The length of the descriptor immediately following this header. The next descriptor header is located exactly Payload_Length bytes from the end of this header i.e. there are no gaps or pad bytes in the Gnutella data stream.</i>

The TTL is the only mechanism for expiring descriptors on the network. Servents should carefully scrutinize the TTL field of received descriptors and lower them as necessary. Abuse of the TTL field will lead to an unnecessary amount of network traffic and poor network performance.

The Payload Length field is the only reliable way for a servant to find the beginning of the next descriptor in the input stream. The Gnutella protocol does not provide an "eye-catcher" string or any other descriptor synchronization method. Therefore, servants should rigorously validate the Payload Length field for each descriptor received (at least for fixed-length descriptors). If a servant becomes out of synch with its input stream, it should drop the connection associated with the stream since the upstream servant is either generating, or forwarding, invalid descriptors.

Immediately following the descriptor header, is a payload consisting of one of the following descriptors:

Ping (0x00)

Ping descriptors have no associated payload and are of zero length. A Ping is simply represented by a Descriptor Header whose Payload_Descriptor field is 0x00 and whose Payload_Length field is 0x00000000.

A servant uses Ping descriptors to actively probe the network for other servants. A servant receiving a Ping descriptor may elect to respond with a Pong descriptor, which contains the address of an active Gnutella servant (possibly the one sending the Pong descriptor) and the amount of data it's sharing on the network.

This specification makes no recommendations as to the frequency at which a servant should send Ping descriptors, although servant implementers should make every attempt to minimize Ping traffic on the network.

Pong (0x01)

	Port	IP Address	Number of Files Shared	Number of Kilobytes Shared
Byte offset	0	1 2	5 6	9 10 13
Port	<i>The port number on which the responding host can accept incoming connections.</i>			
IP Address	<i>The IP address of the responding host.</i>			
	<i>This field is in little endian format.</i>			
Number of Files Shared	<i>The number of files that the servent with the given IP address and port is sharing on the network.</i>			
Number of Kilobytes Shared	<i>The number of kilobytes of data that the servent with the given IP address and port is sharing on the network.</i>			

Pong descriptors are only sent in response to an incoming Ping descriptor. It is valid for more than one Pong descriptor to be sent in response to a single Ping descriptor. This enables host caches to send cached servent address information in response to a Ping request.

Query (0x80)

	Minimum Speed	Search criteria
Byte offset	0	1 2 ...
Minimum Speed	<i>The minimum speed (in kB/second) of servents that should respond to this message. A servent receiving a Query descriptor with a Minimum Speed field of n kB/s should only respond with a QueryHit if it is able to communicate at a speed $\geq n$ kB/s</i>	
Search Criteria	<i>A nul (i.e. 0x00) terminated search string. The maximum length of this string is bounded by the Payload_Length field of the descriptor header.</i>	

QueryHit (0x81)

	Number of Hits	Port	IP Address	Speed	Result Set	Servent Identifier
Byte offset	0	1 2	3 6	7 10	11 ...	n n + 16
Number of Hits	<i>The number of query hits in the result set (see below).</i>					
Port	<i>The port number on which the responding host can accept incoming connections.</i>					
IP Address	<i>The IP address of the responding host.</i>					
Speed	<i>The speed (in kB/second) of the responding host.</i>					

Result Set *A set of responses to the corresponding Query. This set contains Number_of_Hits elements, each with the following structure:*

	File Index			File Size			File Name		
Byte offset	0	3	4	7	8		...		

File Index *A number, assigned by the responding host, which is used to uniquely identify the file matching the corresponding query.*

File Size *The size (in bytes) of the file whose index is File_Index*

File Name *The double-nul (i.e. 0x0000) terminated name of the file whose index is File_Index.*

The size of the result set is bounded by the size of the Payload_Length field in the Descriptor Header.

Servent Identifier *A 16-byte string uniquely identifying the responding servent on the network. This is typically some function of the servent's network address. The Servent Identifier is instrumental in the operation of the Push Descriptor (see below).*

QueryHit descriptors are only sent in response to an incoming Query descriptor. A servent should only reply to a Query with a QueryHit if it contains data that strictly meets the Query Search Criteria.

The Descriptor_Id field in the Descriptor Header of the QueryHit should contain the same value as that of the associated Query descriptor. This allows a servent to identify the QueryHit descriptors associated with Query descriptors it generated.

Push (0x40)

	Servent Identifier				File Index				IP Address				Port	
Byte offset	0	15	16	19	20	23	24	25						

Servent Identifier *The 16-byte string uniquely identifying the servent on the network who is being requested to push the file with index File_Index. The servent initiating the push request should set this field to the Servent_Identifier returned in the corresponding QueryHit descriptor. This allows the recipient of a push request to determine whether or not it is the target of that request.*

File Index *The index uniquely identifying the file to be pushed from the target servent. The servent initiating the push request should set this field to the value of one of the File_Index fields from the Result Set in the corresponding QueryHit descriptor.*

IP Address *The IP address of the host to which the file with File_Index should be pushed.*

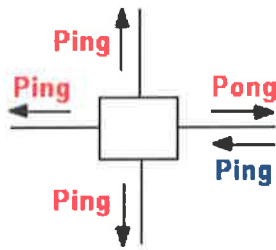
Port *The port to which the file with index File_Index should be pushed.*

A servent may send a Push descriptor if it receives a QueryHit descriptor from a servent that doesn't support incoming connections. This might occur when the servent sending the QueryHit descriptor is behind a firewall. When a servent receives a Push descriptor, it may act upon the push request if and only if the `Servent_Identifier` field contains the value of its servent identifier.

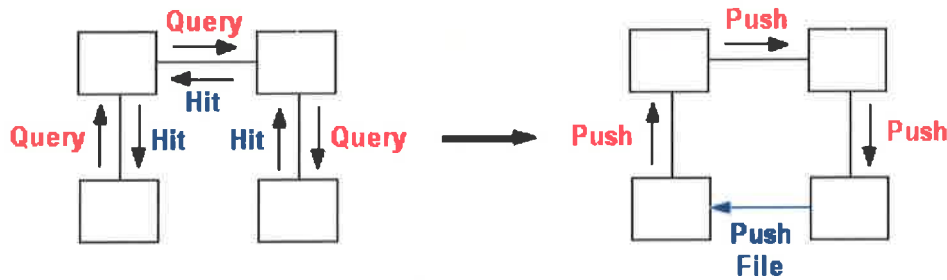
Descriptor Routing

The peer-to-peer nature of the Gnutella network requires servents to route network traffic (queries, query replies, push requests, etc.) appropriately. A well-behaved Gnutella servent will route protocol descriptors according to the following rules:

1. Pong descriptors may only be sent along the same path that carried the incoming Ping descriptor. This ensures that only those servents that routed the Ping descriptor will see the Pong descriptor in response. A servent that receives a Pong descriptor with Descriptor ID = n , but has not seen a Ping descriptor with Descriptor ID = n should remove the Pong descriptor from the network.
2. QueryHit descriptors may only be sent along the same path that carried the incoming Query descriptor. This ensures that only those servents that routed the Query descriptor will see the QueryHit descriptor in response. A servent that receives a QueryHit descriptor with Descriptor ID = n , but has not seen a Query descriptor with Descriptor ID = n should remove the QueryHit descriptor from the network.
3. Push descriptors may only be sent along the same path that carried the incoming QueryHit descriptor. This ensures that only those servents that routed the QueryHit descriptor will see the Push descriptor. A servent that receives a Push descriptor with Descriptor ID = n , but has not seen a QueryHit descriptor with Descriptor ID = n should remove the Push descriptor from the network.
4. A servent will forward incoming Ping and Query descriptors to all of its directly connected servents, except the one that delivered the incoming Ping or Query.
5. A servent will decrement a descriptor header's TTL field, and increment its Hops field, before it forwards the descriptor to any directly connected servent. If, after decrementing the header's TTL field, the TTL field is found to be zero, the descriptor is not forwarded along any connection.
6. A servent receiving a descriptor with the same Payload Descriptor and Descriptor ID as one it has received before, should attempt to avoid forwarding the descriptor to any connected servent. Its intended recipients have already received such a descriptor, and sending it again merely wastes network bandwidth.



Example 1. Ping/Pong Routing



Example 2. Query/QueryHit/Push Routing

File Downloads

Once a servent receives a QueryHit descriptor, it may initiate the direct download of one of the files described by the descriptor's Result Set. Files are downloaded out-of-network i.e. a direct connection between the source and target servent is established in order to perform the data transfer. File data is never transferred over the Gnutella network.

The file download protocol is HTTP. The servent initiating the download sends a request string of the following form to the target server:

```
GET /get/<File Index>/<File Name>/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-\r\n
\r\n
```

where <File Index> and <File Name> are one of the File Index/File Name pairs from a QueryHit descriptor's Result Set. For example, if the Result Set from a QueryHit descriptor contained the entry

File Index	2468
File Size	4356789
File Name	FooBar.mp3\x00\x00

then a download request for the file described by this entry would be initiated as follows:

```
GET /get/2468/FooBar.mp3/ HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-\r\n
\r\n
```


The server receiving this download request responds with HTTP 1.0 compliant headers such as

```
HTTP 200 OK\r\n
Server: Gnutella\r\n
Content-type: application/binary\r\n
Content-length: 4356789\r\n
\r\n
```

The file data then follows and should be read up to, and including, the number of bytes specified in the Content-length provided in the server's HTTP response.

The Gnutella protocol provides support for the HTTP Range parameter, so that interrupted downloads may be resumed at the point at which they terminated.

The Free Network Project - Workshop Paper



The Freenet Project

"Re-Wiring the Internet"

Sections

[-Front Page](#)
[-SiteMap](#)
[-About](#)
[-Freenet News](#)
[-Download](#)
[-Documentation](#)
[-Development](#)
[-Translations](#)

Documentation

[-Creating websites in Freenet](#)
[-Client Writer's Documentation](#)
[-Crypto Layer](#)
[-How Freenet Works](#)
[-Original Paper \[gzip\]](#)
[-Proposed Ideas](#)
[-Installation](#)
[-Javadocs](#)
[-Guidelines for key construction](#)
[-Bestiary of Keys](#)
[-Metadata standards](#)
[-Censorship and Copyright](#)
[-Freenet Protocol](#)
[-0.3 Protocol](#)
[-Key Indexes](#)
[-Workshop Paper](#)
[-Proposed update mechanisms](#)
[-User-contributed documentation](#)
[-Freenet on Win2K](#)
[-Install on Windows](#)

Thanks To



Domains donated by:
[Edward Alfert](#)
 Mousetra

Workshop Paper

The following paper was written by [Theodore Hong](#) and will appear in the [ICSI Workshop on Design Issues in Anonymity and Unobservability](#), July 25-26, Berkeley, California. It is also available in [postscript](#) [\[gzip\]](#).

Freenet: A Distributed Anonymous Information Storage and Retrieval System

Ian Clarke
 School of Artificial Intelligence,
 Division of Informatics,
 University of Edinburgh,
 Edinburgh,
 Scotland.
i.clarke@dynamicblue.com

Brandon Wiley
 College of Communication
 University of Texas at Austin
 Austin, TX 78712
 USA
blanu@uts.cc.utexas.edu

Oskar Sandberg
 Department of Numerical Analysis and Computer
 Science
 Royal Institute of Technology
 SE-100 44 Stockholm
 Sweden
md98-osa@nada.kth.se

Theodore W. Hong¹
 Department of Computing
 Imperial College of Science, Technology and Medicine
 180 Queen's Gate, London SW7 2BZ
 United Kingdom
t.hong@doc.ic.ac.uk

July 1, 2000

Abstract:

We describe Freenet, a peer-to-peer network application that permits the publication, replication, and retrieval of data while protecting the anonymity of both authors and readers. Freenet operates as a network of identical nodes that collectively pool their storage space to store data files, and cooperate to route requests to the most likely physical location of data. No broadcast search or centralized location index is employed. Files are referred to in a location-independent manner, and are dynamically replicated in locations near requestors and deleted from locations where there is no interest. It is infeasible to discover the true origin or destination of a file passing through the network, and difficult for a node operator to determine or be held responsible for the actual physical contents of her own node.

- [Introduction](#)
- [Related work](#)
- [Architecture](#)
 - [Retrieving data](#)
 - [Storing data](#)
 - [Managing data](#)
- [Protocol details](#)
- [Naming, searching, and updating](#)
- [Performance simulation](#)
 - [Retrieval success rate](#)
 - [Retrieval time](#)
- [Security](#)
- [Conclusions](#)
- [Acknowledgements](#)
- [Bibliography](#)
- [About this document ...](#)

Introduction

Computer networks are rapidly growing in importance as a medium for the storage and exchange of information. However, current systems afford little privacy to their users, and typically store any given data item in only one or a few fixed places, creating a central point of failure. Because of a continued desire among individuals to protect the privacy of their authorship or readership of various types of sensitive information[24], and the undesirability of central points of failure which can be attacked by opponents wishing to remove data from the system[10,23] or simply overloaded by too much interest[1], systems offering greater security and reliability are needed.

We are developing Freenet, a distributed information storage and retrieval system designed to address these concerns of privacy and availability. The system operates as a location-independent distributed file system across many individual computers that allows files to be inserted, stored, and requested anonymously. There are five main design goals:

- Anonymity for both producers and consumers of information
- Deniability for storers of information
- Resistance to attempts by third parties to deny access to information
- Efficient dynamic storage and routing of information
- Decentralization of all network functions

The system is designed to respond adaptively to usage patterns, transparently moving, replicating, and deleting files as necessary to provide efficient service without resorting to broadcast searches or centralized location indexes. It is not intended to guarantee permanent file storage, although it is hoped that enough nodes will join with enough storage capacity that most files will be able to remain indefinitely. In addition, the system operates at the application layer and assumes the existence of a secure transport layer, although it is transport-independent. It does not seek to provide anonymity for general network usage, only for Freenet file transactions.

Freenet is currently being developed as a free software project on Sourceforge, and a preliminary implementation can be downloaded from <http://freenet.sourceforge.net/>. It grew out of work originally done by the first author at the University of Edinburgh[11].

Related work

Several strands of related work in this area can be distinguished. Anonymous point-to-point channels based on Chaum's mix-net scheme[7] have been implemented for email by the Mixmaster remailer[19] and for general TCP/IP traffic by onion routing[17] and Freedom[27]. Such channels are not in themselves easily suited to one-to-many publication, however, and are best viewed as a complement to Freenet since they do not provide file access and storage.

Anonymity for consumers of information in the web context is provided by browser proxy services such as the Anonymizer[5], although they provide no protection for producers of information and do not protect consumers against logs kept by the services themselves. Private information retrieval schemes[9] provide much stronger guarantees for information consumers, but only to the extent of hiding which piece of information was retrieved from a particular server. In most cases, the fact of contacting a particular server in itself reveals much about the information retrieved, which can only be counteracted by having each server hold all information (naturally this scales poorly). The closest work to our own is Reiter and Rubin's Crowds system[21], which uses a similar method of proxying requests for consumers, although Crowds does not itself store information and does not protect information producers. Berthold *et al.* propose Web Mixes[6], a stronger system which uses message padding and reordering and dummy messages to increase security, but again does not protect information producers.

The Rewebber[22] provides a measure of anonymity for producers of web information by means of an encrypted URL service which is essentially the inverse of an anonymizing browser proxy, but has the same difficulty of providing no protection against the operator of the service itself. TAZ[16] extends this idea by using chains of nested encrypted URLs which successively point to different rewebber servers to be contacted, although this is vulnerable to traffic analysis using replay. Both rely on a single server as the ultimate source of information. Publius[26] enhances availability by distributing files as redundant shares among n web servers, only k of which are needed to reconstruct a file; however, since the identity of the servers themselves is not anonymized, an attacker might remove information by forcing the closure of $n-k+1$ servers. The Eternity proposal[4] seeks to archive information permanently and anonymously, although it lacks specifics on how to efficiently locate stored files, making it more akin to an anonymous backup service. Free Haven[12] is an interesting anonymous publication system which uses a trust network and file trading mechanism to provide greater server accountability while maintaining anonymity.

distributed.net[13] demonstrated the concept of pooling computer resources among multiple users on a large scale for CPU cycles; other systems which do the same for disk space are Napster[20] and Gnutella[15], although the former relies on a central server to locate files and the latter employs an inefficient broadcast search. Neither one replicates files. Intermemory[8] and India[14] are cooperative distributed filesystem systems intended for long-term archival storage along the lines of Eternity, in which files are split into redundant shares and distributed among many participants. Akamai[2] provides a service which replicates files at locations near information consumers, but is not suitable for producers who are individuals (as opposed to corporations). None of these systems attempt to provide anonymity.

Architecture

Freenet is implemented as a peer-to-peer network of nodes that query one another to store and retrieve data files, which are named by location-independent keys. Each node maintains its own local datastore which it makes available to the network for reading and writing, as well as a dynamic routing table containing addresses of other nodes and the keys that they are thought to hold. It is intended that most users of the system will run nodes, both to provide security guarantees against inadvertently using a hostile foreign node and to increase the storage capacity available to the network as a whole.

The system can be regarded as a cooperative distributed filesystem incorporating location independence and transparent lazy replication. Just as systems such as `distributed.net` [13] enable ordinary users to share unused CPU cycles on their machines, Freenet enables users to share unused disk space. However, where `distributed.net` uses those CPU cycles for its own purposes, Freenet is directly useful to users themselves, acting as an extension to their own hard drives.

The basic model is that queries are passed along from node to node in a chain of proxy requests, with each node making a local routing decision in the style of IP routing about where to send the query next (this varies from query to query). Nodes know only their immediate upstream and downstream neighbors in the chain. Each query is given a "hops-to-live" count which is decremented at each node to prevent infinite chains (analogous to IP's time-to-live). Each query is also assigned a pseudo-unique random identifier, so that nodes can prevent loops by rejecting queries they have seen before. When this happens, the immediately preceding node simply chooses a different node to forward to. This process continues until the query is either satisfied or exceeds its hops-to-live limit. Then, the success or failure result is passed back up the chain to the sending node.

No node is privileged over any other node, so no hierarchy or central point of failure exists. Joining the network is simply a matter of discovering the address of one or more existing nodes through out-of-band means, then starting to send messages. Files are immutable at present, although file updatability is on the agenda for future releases. In addition, the namespace is currently flat, consisting of the space of 160 bit SHA-1 [3] hashes of descriptive text strings, although support for a richer namespace is a priority for development. See section 5 for further discussion of updating and naming.

Retrieving data

To retrieve data, a user hashes a short descriptive string (for example, `text/philosophy/sun-tzu/art-of-war`) to obtain a file key. (See section 5 for details on how descriptive strings corresponding to files can be discovered.) She then sends a request message to her own node specifying that key and a hops-to-live value. When a node receives a request, it first checks its own store for the data and returns it if found, together with a note saying it was the source of the data. If not found, it looks up the nearest key in its routing table to the key requested and forwards the request to the corresponding node. If that request is ultimately successful and returns with the data, the node will pass the data back to the upstream requestor, cache the file in its own datastore, and create a new entry in its routing table associating the actual data source with the requested key. A subsequent request for the same key will be immediately satisfied from the local cache; a request for a "similar" key (determined by lexicographic distance) will be forwarded to the previously successful data source. Because maintaining a table of data sources is a potential security concern, any node along the way can unilaterally decide to change the reply message to claim itself or another arbitrarily-chosen node as the data source.

If a node cannot forward a request to its preferred downstream node because the target is down or a loop would be created, the node having the second-nearest key will be tried, then the third-nearest, and so on. If a node runs out of candidates to try, it reports failure back to its upstream neighbor, which will then try *its* second choice, etc. In this way, a request operates as a steepest-ascent hill-climbing search with backtracking. If the hops-to-live count is exceeded, a failure result is propagated back to the original requestor without any further nodes being tried. Nodes may unilaterally curtail excessive hops-to-live values to reduce network load. They may also forget about pending requests after a period of time to keep message memory free.

Figure 1 depicts a typical sequence of request messages.

```

\begin{figure}\centering
\psfig{figure=icsi-fig1.eps,bllx=145bp,bllly=535bp,bburx=440bp,bbury=675bp,clip=}\end{figure}

```

Figure 1: A typical request sequence.

The user initiates a request at node *a*. Node *a* forwards the request to node *b*, which forwards it to node *c*. Node *c* is unable to contact any other nodes and returns a backtracking "request failed" message to *b*. Node *b* then tries its second choice, *e*, which forwards the request to *f*. Node *f* forwards the request to *b*, which detects the loop and returns a backtracking failure message. Node *f* is unable to contact any other nodes and backtracks one step further back to *e*. Node *e* forwards the request to its second choice, *d*, which has the data. The data is returned from *d* via *e* and *b* back to *a*, which sends it back to the user. The data is also cached on *e*, *b*, and *a*.

This mechanism has a number of effects. Most importantly, we hypothesize that the quality of the routing should improve over time, for two reasons. First, nodes should come to specialize in locating sets of similar keys. If a node is listed in routing tables under a particular key, it will tend to receive mostly requests for keys similar to that key. It is therefore likely to gain more "experience" in answering those queries and become better informed in its routing tables about which other nodes carry those keys. Second, nodes should become similarly specialized in storing clusters of files having similar keys. Because forwarding a request successfully will result in the node itself gaining a copy of the requested file, and most requests will be for similar keys, the node will mostly acquire files with similar keys. Taken together, these two effects should improve the efficiency of future requests in a self-reinforcing cycle, as nodes build up routing tables and datastores focusing on particular sets of keys, which will be precisely those keys that they are asked about.

In addition, the request mechanism will cause popular data to be transparently replicated by the system and mirrored closer to requestors. For example, if a file that is originally located in London is requested in Berkeley, it will become cached locally and provide faster response to subsequent Berkeley requests. It also becomes copied onto each computer along the way, providing redundancy if the London node fails or is shut down. (Note that "along the way" is determined by key closeness and does not necessarily have geographical relevance.)

Finally, as nodes process requests, they create new routing table entries for previously-unknown nodes that supply files, increasing connectivity. This helps new nodes to discover more of the network (although it does not help the rest of the network to discover *them*; for that, performing inserts is necessary). Note that direct links are created, bypassing the intermediate nodes used. Thus, nodes that successfully supply data will gain routing table entries and be contacted more often than nodes that do not.

Because hashes are used as keys, lexicographic closeness of keys does not imply any closeness of the original descriptive strings and presumably, no closeness of subject matter of the files. This lack of semantic closeness is not important, however, as the routing algorithm is based on knowing where keys are located, not where subjects are located. That is, supposing `text/philosophy/sun-tzu/art-of-war` hashes to AH5JK2, requests for this file can be routed more effectively by creating clusters containing AH5JK1, AH5JK2, and AH5JK3, not by creating clusters for works of philosophy. Indeed, the use of a hash is desirable precisely because philosophical works will be scattered across the network, lessening the chances that failure of a single node will make all philosophy unavailable.

Storing data

Inserts follow a parallel strategy to requests. To insert data, a user picks an appropriate descriptive text string and hashes it to create a file key. She then sends an insert message to her own node specifying the proposed key and a hops-to-live value (this will determine the number of nodes to store it on). When a node receives an insert proposal, it first checks its own store to see if the key is already taken. If the key is found, the node returns the pre-existing file as if a request had been made for it. The user will thus know that a collision was encountered and can try again using a different key, i.e. different descriptive text. (Although potentially the use of a hash might cause extra collisions, in practice a high-quality hash of sufficient length should only cause collisions if the same initial descriptive text was chosen.) If the key is not found, the node looks up the nearest key in its routing table to the key proposed and forwards the insert to the corresponding node. If that insert causes a collision and returns with the data, the node will pass the data back to the upstream inserter and again behave as if a request had been made (i.e. cache the file locally and create a routing table entry for the data source).

If the hops-to-live limit is reached without a key collision being detected, an "all clear" result will be propagated back to the original inserter. Note that for inserts, this is a successful result, in contrast to the request case. The user then sends the data to insert, which will be propagated along the path established by the initial query and stored in

EXHIBIT A

each node along the way. Each node will also create an entry in its routing table associating the inserter (as the data source) with the new key. To avoid the obvious security problem, any node along the way can unilaterally decide to change the insert message to claim itself or another arbitrarily-chosen node as the data source.

If a node cannot forward an insert to its preferred downstream node because the target is down or a loop would be created, the insert backtracks to the second-nearest key, then the third-nearest, and so on in the same way as for requests. If the backtracking returns all the way back to the original inserter, it indicates that fewer nodes than asked for could be contacted. As with requests, nodes may curtail excessive hops-to-live values and/or forget about pending inserts after a period of time.

This mechanism has three effects. First, newly inserted files are selectively placed on nodes already possessing files with similar keys. This reinforces the clustering of keys set up by the request mechanism. Second, new nodes can tell the rest of the network of their existence by inserting data. Third, an attempt by an attacker to supplant an existing file by deliberate insert collision (e.g., by inserting a corrupted or empty file under the same key) is likely to simply spread the real file further, since the original file is propagated back on collision. (The use of a content-hash key, described in section 5, makes such an attack infeasible altogether.)

Managing data

All information storage systems must deal with the problem of finite storage capacity. Individual Freenet node operators can configure the amount of storage to dedicate to their datastores. Node storage is managed as an LRU (Least Recently Used) cache[25], with data items kept sorted in decreasing order by time of most recent request (or time of insert, if an item has never been requested). When a new file arrives (from either a new insert or a successful request) which would cause the datastore to exceed the designated size, the least recently used files are evicted in order until there is room. The impact on availability is mitigated somewhat by the fact that the routing table entries created when the evicted files first arrived will remain for a time, potentially allowing the node to later get new copies from the original data sources. (Routing table entries are also eventually deleted in a similar fashion as the table fills up, although they will be retained longer since they are smaller.)

Strictly speaking, the datastore is not a cache, since the set of datastores is all the storage that there is, i.e. there is no "permanent" copy which is being duplicated. Once all the nodes have decided, collectively speaking, to drop a particular file, it will no longer be available to the network. In this respect, Freenet differs from the Eternity service, which seeks to provide guarantees of file lifetime.

This mechanism has an advantageous side, however, since it allows outdated documents to fade away after being superseded by newer documents, thus alleviating some of the problem of immutable files. If an outdated document is still used and considered valuable for historical reasons, it will stay alive precisely as long as it continues to be requested.

For political or legal reasons, it may be desirable for node operators not to explicitly know the contents of their datastores. Therefore, it is recommended that all inserted files be encrypted by their original unhashed descriptive text strings in order to obscure their contents. Of course, this does not secure the file--that would be impossible since a requestor (potentially anyone) must be capable of decrypting the file once retrieved. Rather, the objective is that the node operator can plausibly deny any knowledge of the contents of her datastore, since all she knows *a priori* is the hashed key and its associated encrypted file. The hash cannot feasibly be reversed to reveal the unhashed description and decrypt the file. With effort, of course, a dictionary attack will reveal which keys are present--as it must in order for requests to work at all.

Protocol details

The Freenet protocol is packet-oriented and uses self-contained messages. Each message includes a transaction ID so that nodes can track the state of inserts and requests. This design is intended to permit flexibility in the choice of transport mechanisms for messages, whether they be TCP, UDP, or other technologies such as packet radio. For efficiency, nodes are also able to send multiple messages over a persistent channel such as a TCP connection, if available. Node addresses consist of a transport method plus a transport-specific identifier (such as an IP address and port number), e.g. `tcp/192.168.1.1:19114`.

A Freenet transaction begins with a Request.Handshake message from one node to another, specifying the desired return address of the sending² node. (The return address may be impossible to determine from the transport layer alone, or may use a different transport from that used to send the message.) If the remote node is active and responding to requests, it will reply with a Reply.Handshake specifying the protocol version number that it understands. Handshakes are remembered for a few hours, and subsequent transactions between the same nodes during this time may omit this step.

All messages contain a randomly-generated 64-bit transaction ID, a hops-to-live counter, and a depth counter. Although the ID cannot be guaranteed to be unique, the likelihood of a collision occurring during the transaction lifetime among the limited set of nodes that it sees is extremely low. Hops-to-live is set by the originator of a message and is decremented at each hop to prevent messages being forwarded indefinitely. To reduce the

EXHIBIT A

information that an attacker can obtain from the hops-to-live value, messages do not automatically terminate after hops-to-live reaches 1 but are forwarded on with finite probability (with hops-to-live again 1). Depth is incremented at each hop and is used by a replying node to set hops-to-live high enough to reach a requestor. Requestors should initialize it to a small random value to obscure their location. As with hops-to-live, a depth of 1 is not automatically incremented but is passed unchanged with finite probability.

To request data, the sending node sends a Request.Data message specifying a transaction ID, initial hops-to-live and depth, and a search key. The remote node will check its datastore for the key and if not found, will forward the request to another node as described in section 3.1. Using the chosen hops-to-live count, the sending node starts a timer for the expected amount of time it should take to contact that many nodes, after which it will assume failure. While the request is being processed, the remote node may periodically send back Reply.Restart messages indicating that messages were stalled waiting on network timeouts, so that the sending node knows to extend its timer.

If the request is ultimately successful, the remote node will reply with a Send.Data message containing the data requested and the address of the node which supplied it (possibly faked). If the request is ultimately unsuccessful and its hops-to-live are completely used up trying to satisfy it, the remote node will reply with a Reply.NotFound. The sending node will then decrement the hops-to-live of the Send.Data (or Reply.NotFound) and pass it along upstream, unless it is the actual originator of the request. Both of these messages terminate the transaction and release any resources held. However, if there are still hops-to-live remaining, usually because the request ran into a dead end where no viable non-looping paths could be found, the remote node will reply with a Request.Continue giving the number of hops-to-live left. The sending node will then try to contact the next-most likely node from its routing table. It will also send a Reply.Restart upstream.

To insert data, the sending node sends a Request.Insert message specifying a randomly-generated transaction ID, an initial hops-to-live and depth, and a proposed key. The remote node will check its datastore for the key and if not found, forward the insert to another node as described in section 3.2. Timers and Reply.Restart messages are also used in the same way as for requests.

If the insert ultimately results in a key collision, the remote node will reply with either a Send.Data message containing the existing data or a Reply.NotFound (if existing data was not actually found, but routing table references to it were). If the insert does not encounter a collision, yet runs out of nodes with nonzero hops-to-live remaining, the remote node will reply with a Request.Continue. In this case, Request.Continue is a failure result meaning that not as many nodes could be contacted as asked for. These messages will be passed along upstream as in the request case. Both messages terminate the transaction and release any resources held. However, if the insert expires without encountering a collision, the remote node will reply with a Reply.Insert, indicating that the insert can go ahead. The sending node will pass along the Reply.Insert upstream and wait for its predecessor to send a Send.Insert containing the data. When it receives the data, it will store it locally and forward the Send.Insert downstream, concluding the transaction.

Naming, searching, and updating

Freenet's basic flat namespace has obvious disadvantages in terms of discovering documents, name collisions, etc. Several mechanisms of providing more structure within the current scheme are possible. For example, directory-like documents containing hypertext pointers to other files could be created. A directory file under the key `text/philosophy` could contain a list of keys such as `text/philosophy/sun-tzu/art-of-war`, `text/philosophy/confucius/analects`, and `text/philosophy/nozick/anarchy-state-utopia`, using appropriate syntax interpretable by a client. The difficulty, however, is in deciding how to permit changes to the directory to update entries, while preventing the directory from being corrupted or spammed. An alternative mechanism is to encourage individuals to maintain and share their own compilations of keys--subjective bookmark lists, rather than authoritative directories. This is the approach in common use on the world-wide web.

Name collisions are still a problem with both bookmark lists and directories. One way of addressing collisions is to introduce a two-level structure similar to that used by most traditional file systems. Real files could be stored under a pseudo-unique binary key, such as a hash of the files' contents (a *content-hash* key). Users would access these files by first retrieving an indirect file stored under a semantically meaningful name. The indirect file would consist solely of a list of binary keys corresponding to that name (possibly only one), along with other information useful for differentiating among the possible choices, such as author, creation time, and endorsements by other users. Content-hash keys would also protect against files being maliciously tampered with or replaced. However, indirect files are essentially like low-level directories, and share the same problem of managing updates. Another approach is to skip the indirect files altogether and have bookmark lists pointing to content-hash keys, rather than names.

Introducing a search capability in conjunction with binary keys offers a way to side-step the need to maintain directories. The most straightforward way to add search is to run a hypertext spider such as those used to search the web. While an attractive solution in many ways, this conflicts with the design goal of avoiding centralization. A possible alternative is to create a special class of lightweight indirect files. When a real file is inserted, the author could also insert a number of indirect files containing a single pointer to the real file, named according to search keywords chosen by her. These indirect files would differ from normal files in that collisions would be permitted on insert, and requests for an indirect file key (i.e. a keyword) would keep going until a specified number of indirect

files (i.e. search results) were accumulated. Managing the likely large volume of these indirect files is an open problem.

Updates by a single user can be handled in a reasonably straightforward manner by using a variation on indirection. When an author inserts a file which she later intends to update, she first generates a public-private key pair and signs the file with the private key. The file is inserted under a binary key, but instead of using the hash of the file's contents, the literal public key itself is used (a *signature-verifying key*). As with inserting under a content-hash key, inserting under a signature-verifying key provides a pseudo-unique binary key for a file, which can also be used to verify that the file's contents have not been tampered with. To update the file, the new version is signed by the private key and inserted under the public signature-verifying key. When the insert reaches a node which possesses the old version, a key collision will occur. The node will check the signature on the new version, verify that it is both valid and more recent, and replace the old version.

In order to allow old versions of files to remain in existence for historical reasons and to prevent the possibility of authors being compelled to "update" their own files out of existence, an additional level of indirection can be used. In this scheme, the real file is inserted under its content-hash key. Then, an indirect file containing a pointer to that content-hash key is created, but inserted under a signature-verifying key. This signature-verifying key is given out as the binary key for the file and entered, for example, in directories and bookmark lists (making them double indirect files). When the author creates a new version, it is inserted under its own content-hash key, distinct from the old version's key. The signature-verified indirect file is then updated to point to the new version (or possibly to keep pointers to all versions). Thus the signature-verifying key will always lead to the most recent version of the file, while old versions can continue to be accessed by content-hash key if desired. (If not requested, however, these old versions will eventually disappear like any other unused file.)

For large files, splitting files into multiple parts is desirable because of storage and bandwidth limitations. Splitting even medium files into standard-sized parts (e.g. 16 kilobytes) also has advantages in combating traffic analysis. This is easily accomplished by inserting each part separately under a content-hash key, and including pointers to the other parts.

Combining all these ideas together, a user might look through a bookmark list or perform a search on a keyword to get a list of signature-verifying binary keys for files dealing with a particular topic. Retrieving one of these keys gives an indirect file containing a set of content-hash keys corresponding to different versions, ordered by date. Retrieving the most recent content-hash key gives the first part of a multipart file, together with pointers to two other content-hash keys. Finally, retrieving those last two content-hash keys and concatenating the three parts together yields the desired file.

Performance simulation

Simulations were carried out on an early version of this system to give some indications about its performance. Here we summarize the most important results; for full details, see [11].

The scenario for these simulations was a network with between 500 and 900 nodes. Each node had a datastore size of 40 items and a routing table size of 50 addresses (relatively low, because of limitations on available hardware), and was given 10 unique items to store locally. The network was initially connected in a linear fashion, where each node started with routing references to one node on either side.

Retrieval success rate

Queries for random keys were sent to random nodes in the network, in batches of 50 parallel queries at a time, and the percentage of successful requests recorded over time. Figure 2 shows the percentage of successful requests versus the total number of queries since initialization, for several network sizes.



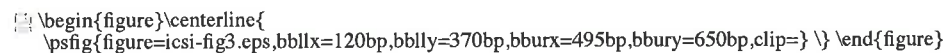
```
\begin{figure}\centerline{
\psfig{figure=icsi-fig2.eps,bblx=120bp,bblly=400bp,bburx=495bp,bbury=680bp,clip=} }\end{figure}
```

Figure 2: Percentage of successful requests over time.

We can see that the initially low success rate rises rapidly until over 95% of requests are successful. The number of queries until network convergence is approximately half the total size of the network.

Retrieval time

Queries for random keys were sent to random nodes in the network, in batches of 50 parallel queries at a time, and the average number of hops needed for a successful request recorded over time. Figure 3 shows the number of request hops versus the total number of queries since initialization, for several network sizes.



```
\begin{figure}\centerline{
\psfig{figure=icsi-fig3.eps,bblx=120bp,bblly=370bp,bburx=495bp,bbury=650bp,clip=} }\end{figure}
```

Figure 3: Number of hops per request over time.

We can see that the initially high number of hops required drops until it stabilizes at approximately 10 hops. This value changes remarkably little with network size, suggesting that the time required for requests should scale quite well. The number of queries until network convergence is approximately equal to the total size of the network.

Security

The primary goal for Freenet security is protecting the anonymity of requestors and inserters of files. It is also important to protect the identity of storers of files. Although trivially anyone can turn a node into a storer by requesting a file through it, thus “identifying” it as a storer, what is important is that there remain other, unidentified, holders of the file so that an adversary cannot remove a file by attacking all of the nodes that hold it. Files must be protected against malicious modification, and finally, the system must be resistant to denial-of-service attacks.

Reiter and Rubin[21] present a useful taxonomy of anonymous communication properties on three axes. The first axis is the type of anonymity: sender anonymity or receiver anonymity, which mean respectively that an adversary cannot determine either who originated a message, or to whom it was sent. The second axis is the adversary in question: a local eavesdropper, a malicious node or collaboration of malicious nodes, or a web server (not applicable to Freenet). The third axis is the degree of anonymity, which ranges from absolute privacy (the presence of communication cannot be perceived) to beyond suspicion (the sender appears no more likely to have originated the message than any other potential sender), probable innocence (the sender is no more likely to be the originator than not), possible innocence, exposed, and provably exposed (the adversary can prove to others who the sender was).

As Freenet communication is not directed towards specific receivers, receiver anonymity is more accurately viewed as key anonymity, that is, hiding the key which is being requested or inserted. Unfortunately, since routing depends on knowledge of the key, key anonymity is not possible in the basic Freenet scheme (but see the discussion of “pre-routing” below). The use of hashes as keys provides a measure of obscurity against casual eavesdropping, but is of course vulnerable to a dictionary attack since unhashed keys must be widely known in order to be useful.

Freenet's anonymity properties under this taxonomy are shown in Table 1.

Table 1: Anonymity properties of Freenet.

System	Attacker	Sender anonymity	Key anonymity
Basic Freenet	local eavesdropper	exposed	exposed
	collaborating nodes	beyond suspicion	exposed
Freenet + pre-routing	local eavesdropper	exposed	beyond suspicion
	collaborating nodes	beyond suspicion	exposed

Against a collaboration of malicious nodes, sender anonymity is preserved beyond suspicion since a node in a request path cannot tell whether its predecessor in the path initiated the request or is merely forwarding it. [21] describes a probabilistic attack which might compromise sender anonymity, using a statistical analysis of the probability that a request arriving at a node *a* is forwarded on or handled directly, and the probability that *a* chooses a particular node *b* to forward to. This analysis is not immediately applicable to Freenet, however, since request paths are not constructed probabilistically. Forwarding depends on whether or not *a* has the requested data in its datastore, rather than chance. If a request is forwarded, the routing tables determine where it is sent to, and could be such that *a* forwards every request to *b*, or never forwards any requests to *b*, or anywhere in between. Nevertheless, the depth value may provide some indication as to how many hops away the originator was, although this is obscured by the random selection of an initial depth and the probabilistic means of incrementing it (see section 4). Similar considerations apply to hops-to-live. Further investigation is required to clarify these issues.

Against a local eavesdropper there is no protection on messages between the user and the first node contacted. Since the first node contacted can act as a local eavesdropper, it is recommended that the user only use a node on her own machine as the first point of entry into the Freenet network. Messages between nodes are encrypted against local eavesdropping, although traffic analysis may still be performed (e.g. an eavesdropper may observe a message going out without a previous message coming in and conclude that the target originated it).

Key anonymity and stronger sender anonymity can be achieved by adding mix-style “pre-routing” of messages. In this scheme, basic Freenet messages are encrypted by a succession of public keys which determine the route that the encrypted message will follow (overriding the normal routing mechanism). Nodes along this portion of the route are unable to determine either the originator of the message or its contents (including the request key), as per the mix-net anonymity properties. When the message reaches the endpoint of the pre-routing phase, it will be injected into the normal Freenet network and behave as though the endpoint were the originator of the message.

Protection for data sources is provided by the occasional resetting of the data source field in replies. The fact that a node is listed as the data source for a particular key does not necessarily imply that it actually supplied that data, or was even contacted in the course of the request. It is not possible to tell whether the downstream node provided the file or was merely forwarding a reply sent by someone else. In fact, the very act of successfully requesting a file places it on the downstream node if it was not already there, so a subsequent examination of that node on suspicion reveals nothing about the prior state of affairs, and provides a plausible legal ground that the data was not there until the act of investigation placed it there. Requesting a particular file with a hops-to-live of 1 does not directly reveal whether or not the node was previously storing the file in question, since nodes continue to forward messages having hops-to-live of 1 with finite probability. The success of a large number of requests for related files, however, may

provide grounds for suspicion that those files were being stored there previously.

Modification or outright replacement of files by a hostile node is an important threat, and not only because of the corruption of the file itself. Since routing tables are based on replies to requests, a node might attempt to steer traffic towards itself by pretending to have files when it does not and simply returning fictitious data. For data stored under content-hash keys or signature-verifying keys, this is not feasible since inauthentic data can be detected unless a node finds a hash collision or successfully forges a cryptographic signature. Data stored under ordinary descriptive text keys, however, is vulnerable. Some protection is afforded by the expectation that files will be encrypted by the descriptive text, since the node must respond to a hashed key request with data encrypted by the original text key, but a dictionary attack is possible using a table of text keys and their hashes. Even partial dictionaries cause problems since the node can behave normally when an unknown key is requested and forge data when the key is known.

Finally, a number of denial-of-service attacks can be envisioned. The most significant threat is that an attacker will attempt to fill all of the network's storage capacity by inserting a large number of garbage files. An interesting possibility for countering this attack is the Hash Cash scheme[18]. Essentially, the scheme requires the inserter to perform a lengthy computation as "payment" before an insert is accepted, thus slowing down an attack. Another alternative is to divide the datastore into two sections, one for new inserts and one for "established" files (defined as files having received at least a certain number of requests). New inserts can only displace other new inserts, not established files. In this way a flood of garbage inserts might temporarily paralyze insert operations but would not displace existing files. It is difficult for an attacker to artificially legitimize her own (garbage) files by requesting them many times since her requests will be satisfied by the first node to hold the data and not proceed any further. She cannot send requests directly to the other downstream nodes holding her files since their identities are hidden from her. However, adopting this scheme may make it difficult for genuine new inserts to survive long enough to be requested by others and become established.

Attackers may attempt to replace existing files by inserting alternate versions under the same keys. Such an attack is not possible against a content-hash key or signature-verifying key, since it requires finding a hash collision or successfully forging a cryptographic signature. An attack against a descriptive text key, on the other hand, may result in both versions coexisting in the network. The way in which nodes react to insert collisions (detailed in section 3.2) is intended to make such attacks more difficult. The success of a replacement attack can be measured by the ratio of corrupt versus genuine versions resulting in the system. However, the more corrupt copies the attacker attempts to circulate (by setting a higher hops-to-live on insert), the greater the chance that an insert collision will be encountered, which would cause an increase in the number of genuine copies.

Conclusions

The Freenet network provides an effective means of anonymous information storage and retrieval. By using cooperating nodes spread over many computers in conjunction with an efficient routing algorithm, it keeps information anonymous and available while remaining highly scalable. Initial deployment of a test version is underway, and is so far proving successful, with over 15,000 copies downloaded and many interesting files in circulation. Because of the nature of the system, it is impossible to tell exactly how many users there are or how well the insert and request mechanisms are working, but anecdotal evidence is so far positive. We are working on implementing a simulation and visualization suite which will enable more rigorous tests of the protocol and routing algorithm. More realistic simulation is necessary which models the effects of inserts taking place alongside requests, nodes joining and leaving, variation in node capacity, and larger network sizes.

Acknowledgements

Portions of this material are based upon work supported under a National Science Foundation Graduate Research Fellowship.

Bibliography

1. S. Adler, "The Slashdot effect: an analysis of three Internet publications," <http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html> (2000).
2. Akamai, <http://www.akamai.com/> (2000).
3. American National Standards Institute, Draft: American National Standard X9.30-199X: *Public-Key*

EXHIBIT A

Cryptography Using Irreversible Algorithms for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA). American Bankers Association (1993).

4. R.J. Anderson, "The Eternity service," in *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology (PRAGOCRYPT '96)*, Prague, Czech Republic (1996).
5. Anonymizer, <http://www.anonymizer.com/> (2000).
6. O. Berthold, H. Federrath, and M. Köhnstopp, "Project `Anonymity and unobservability in the Internet'," in *Computers Freedom and Privacy Conference 2000 (CFP 2000) Workshop on Freedom and Privacy by Design* (to appear).
7. D.L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM* **24**(2), 84-88 (1981).
8. Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos, "A prototype implementation of archival intermemory," in *Proceedings of the Fourth ACM Conference on Digital Libraries (DL '99)*, Berkeley, CA, USA. ACM Press: New York (1999).
9. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," *Journal of the ACM* **45**(6), 965-982 (1998).
10. Church of Spiritual Technology (Scientology) v. Dataweb *et al.*, Cause No. 96/1048, District Court of the Hague, The Netherlands (1999).
11. I. Clarke, "A distributed decentralised information storage and retrieval system," unpublished report, Division of Informatics, University of Edinburgh (1999). Available at <http://freenet.sourceforge.net/> (2000).
12. R.R. Dingleline, "The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven," M.Eng. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2000). Available at <http://www.freehaven.net/> (2000).
13. Distributed.net, <http://www.distributed.net/> (2000).
14. D.J. Ellard, J.M. Megquier, and L. Park, "The INDIA protocol," <http://www.eecs.harvard.edu/~ellard/India-www/> (2000).
15. Gnutella, <http://gnutella.wego.com/> (2000).
16. I. Goldberg and D. Wagner, "TAZ servers and the rewebber network: enabling anonymous publishing on the world wide web," <http://www.cs.berkeley.edu/~daw/classes/cs268/taz-www/rewebber.html> (2000).
17. D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private Internet connections," *Communications of the ACM* **42**(2), 39-41 (1999).
18. Hash Cash, <http://www.cypherspace.org/~adam/hashcash/> (2000).
19. Mixmaster, <http://www.obscura.com/~loki/remailer/mixmaster-faq.html> (2000).
20. Napster, <http://www.napster.com/> (2000).
21. M.K. Reiter and A.D. Rubin, "Anonymous web transactions with Crowds," *Communications of the ACM* **42**(2), 32-38 (1999).
- 22.

The Rewebber, <http://www.rewebber.de/> (2000).

23. M. Richtel and S. Robinson, "Several web sites are attacked on day after assault shut Yahoo," *The New York Times*, February 9, 2000.
24. J. Rosen, "The eroded self," *The New York Times*, April 30, 2000.
25. A.S. Tanenbaum, *Modern Operating Systems*. Prentice-Hall: Upper Saddle River, NJ, USA (1992).
26. M. Waldman, A.D. Rubin, and L.F. Cranor, "Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system," in *Ninth USENIX Security Symposium*, Denver, CO, USA (to appear).
27. Zero-Knowledge Systems, <http://www.zks.net/> (2000).

Footnotes

- ... Hong¹
Supported by grants from the Marshall Aid Commemoration Commission and the National Science Foundation.
- ... sending²
Note that the sending node may not be the original requestor.

About this document ...

Freenet: A Distributed Anonymous Information Storage and Retrieval System

This document was generated using the [LaTeX2HTML](#) translator Version 99.2beta6 (1.42)

Copyright © 1993, 1994, 1995, 1996, [Nikos Drakos](#), Computer Based Learning Unit, University of Leeds.
Copyright © 1997, 1998, 1999, [Ross Moore](#), Mathematics Department, Macquarie University, Sydney.

The command line arguments were:

`latex2html -split 0 -nofootnode -nonavigation -local_icons icsi.tex`

The translation was initiated by Theodore Hong on 2000-07-04

[Theodore Hong](#)

Not Found

The requested URL /icsi.ps was not found on this server.

Apache/1.3.26 Server at freenet.sourceforge.net Port 80



Cap'n Bry's gnutella search

Search for: Server timeout 60 secs  

- ▶ [Problems](#)
- ▶ [Identification?](#)
- ▶ [Links](#)
- ▶ [The protocol](#)
- ▶ [Screenshots](#)
- ▶ [My story](#)
- ▶ [PHP source](#)
- ▶ [Search state](#)

Why?

CapnBry's PHP Gnutella Search is a simple script written in [PHP 4](#). It was created on a whim because I was tired of doing paying work one day, gnutella had just come out, and I had just finished enough reverse-engineering of the protocol to do something silly with it.

Licensing

Full licensing details are available from the main php file (gs.php). Here is the exciting bits:

License to copy and use this software is granted provided that it is identified as the "CapnBry's PHP gnutella search" in all material mentioning or referencing this software or this function. License is also granted to make and use derivative works provided that such works are identified as "derived from CapnBry's PHP gnutella search" in all material mentioning or referencing the derived work.

The overall gist is that you can do whatever the hell you want with my code (run a copy of it, build on it) as long as you provide credit to the author, *me*. There are a few ways to do this depending on how nice you want to be:

- **Really nice** - Place a "based on CapnBry's PHP gnutella search" line somewhere on your search page. If you wanted to make me really happy, link it back to this site.
- **Moderately nice** - Insert a comment in the commonHeader function before the search form. Something to the effect of: `<!-- Based on CapnBry's PHP gnutella search -->` would be pretty sweet.
- **Just plain nice** - Include the PHP source code for the search somewhere on your site, and just leave the original license in it.

Network Architecture

The script connects to a real, live gnutella client running on another machine to do its searches. This provides the fan-out, since the PHP script only participates as a leaf node in the gnutellanet.

The Files

The most recent version of the script I've broken into several peices, since the old code was starting to get unmanagable all in one file. Here's what I got:

- [gs.php](#) - This is the main source file it contains all of the code for dealing with the gnutellanet, and php requires files below.
- [gs_config.php](#) - The configuration used by all units. Includes database and gnutellanet connection information. This is where to start hacking the script to get it running on your box.
- [top_n_bottom.php](#) - The common header and footers for the gnutella search pages. I dunno who didn't add #include to the original HTTP spec, but they should be shot.
- [index_text.php](#) - This is the text which appears in the main frame when no search is requested. Kinda like an index.html.
- [status_window.php](#) - Functions for manipulating that groovy status window that pops up while you're doing your search. Sends JavaScript to the client browser.

EXHIBIT A

- [error_msgs.php](#) - Sometimes the machine running the host gnutella gets rebooted or crashes. These are just some pretty error messages for when that happens.
 - [byte_funcs.php](#) - Since there are no higher level functions for reading words, integers, and IP addresses from socket descriptors in PHP (as far as I know), I wrote some of my own. A cheesy CoCreateGUID is also included, and I know, it's not a *real* Microsoft GUID. Close enough for our purposes.
 - [search_history.php](#) - Functions for saving who searches for what, when, and how many results they get. Used to build the stats and "Last 10 Searches" stuff. Requires a MySQL database connection.
 - [gs_mysql.sql](#) - The MySQL script to create the gnutella database used by the search history routines. This may or may not be up to date with what's in my database, so if you see an inconsistency, lemme know and I'll dump the schema again.
 - [links.php](#) - Just the list of links. Not required, but you'll have to remove the "links" link from the commonHeader().
-
- [gs_old.php](#) - If you're not interested in creating a MySQL database and putting all these files on your server, here is the old source which is all in one file and just uses a flat file to store the last 10 searches. This is unmaintained code at this point, so don't expect me to be adding features or bug fixes to it.

[Email CapnBry](#) *remove spam from the address for it to work*

[Home](#) | [Other things on linux](#)

CapnBry's PHP Gnutella search v0.4 - See source code for licensing information



Cap'n Bry's gnutella search

Search for: Server timeout 60 secs  

- [Problems downloading?](#)
- [Links](#)
- [The protocol](#)
- [Screenshots](#)
- [My clone](#)
- [PHP source](#)
- [Search stats](#)

Trouble Downloading?

The most common problems I've heard about stem from the fact that a gnutella server does not send a Content-type: HTTP header field. This means that when the browser starts to receive the file, it doesn't know what it should do with it, so it just **displays binary data**. If this happens to you, you have three options. The first is just to right-click the link and select "Save Target As..." (in IE) or "Save link as..." (in Netscape). Your other option is to configure your browser to know what to do with those file types. I've included instructions for how to do that below. Your final, and coolest option is to sign up for a myplay.com locker. They'll provide you with 3 GB of online storage space for your mp3 files, and what's best, they'll even download the files for you and put them in your locker. Just click the "add to locker" link!

Internet Explorer

1. Click the Start button in the lower-left corner of the screen.
2. Go to Settings, then Folder Options.
3. Select File Types, and then scroll down until you find "Winamp media file" and "Winamp playlist file."
4. Click "Winamp media file," then click the Edit button. The settings should read as follows:
 - Content Type (MIME): audio/mpeg
 - Default Extension for Content Type: .mp3
 - Actions: Play in Winamp (in bold)
5. Click "Winamp playlist file," then click the Edit button. The settings should read as follows:
 - Content Type (MIME): audio/x-mpegurl
 - Default Extension for Content Type: .m3u
 - Actions: Play in Winamp (in bold)

NOTE: Follow the same steps for Sonique.

Netscape Navigator

1. Click Edit on the toolbar of Netscape.
2. Select Preferences.
3. In Category, select Navigator, then Applications.
4. Click the New Type button and enter the following settings:
 - Description of Type: MP3 file
 - File extension: MP3
 - MIME Type: audio/mpeg
 - Application to use: (Browse to the WinAmp or Sonique .exe file on your computer)
 Check the default box, and then click OK.
5. Click the New Type button again and enter the following settings:
 - Description of Type: MP3 file
 - File extension: M3U
 - MIME Type: audio/x-mpegurl
 - Application to use: (Browse to the WinAmp or Sonique .exe file on your computer)
 Check the default box, and then click OK.

[Email CapnBry](#) remove spam from the address for it to work

[Home](#) | [Other things on linux](#)

CapnBry's PHP Gnutella search v0.4 - See source code for licensing information



Cap'n Bry's gnutella search

Search for:

Server timeout

- Problems
- atomized
- Links
- The website
- Stability
- My clone
- PHP source
- Search stats

The Connections Screen

gnugnutella v0.00 - Built Today on yesterday's technology

gnutellaNet

Uploads

Downloads

Search

Monitor

Config

Connections:

1 gnutellaNet

0 uploads

0 downloads

Buttons

gnutellaNet stats:

0 hosts

0 files

0 MB

0ms avg ping

Update

gnutellaNet connections:

Host	Type	Info (W,R,D)
127.0.0.1:22674	Outgoing	Connected

Remove

Add

Try to keep at least connections up.

Network Information

gnutellaNet Messages:

Searches to local DB:

Routing errors:

Dropped messages:

Upload count (files):

Download count (files):

gnutellaNet host catcher:

24.218.207.88:6346

24.8.41.89:6346

130.160.204.44:6346

216.226.31.193:6346

208.211.49.78:6346

24.95.40.126:6346

207.158.139.5:6346

Connect

Clear

Remove

Get More

☒ Run Log

Copyright (C) 2000 - This is a complete ripoff from Nullsoft, Inc

Search Criteria: blowjob

Message ID: (77F4F5A8-E5D0-43AD-B05E-7D308E184656)

Function ID: 0x00 (Query)

TTL Remaining: 5

Hops Taken: 2

Data Len: 6

Query:

Minimum Speed: 0

Search Criteria: fm4

Message ID: (DAC924A9-FCE2-11D3-89DF-0090279D45E3)

Function ID: 0x00 (Init)

TTL Remaining: 2

Hops Taken: 3

Data Len: 0

Init message in, sending response...

The Search Screen



The Search Monitor



Email Cap'n Bry remove spam from the address for it to work

Home | Other things on linux

Cap'n Bry's PHP Gnutella search v0.4 - See source code for licensing information

Not Found

The requested URL /gnutella/protocol.php was not found on this server.

Apache/2.2.3 (Ubuntu) PHP/5.2.1 Server at capnbry.dyndns.org Port 80



Cap'n Bry's gnutella search

Search for: Server timeout 60 secs 

Gnutella in action

- Problems
- downloading?
- Links
- The endpoint
- Screenshots
- My clone
- PHP source
- Search stats

Gnutella v0.48 Mar 13 2000 19:03:46

gnutellaNet

- Uploads
- Downloads**
- Search
- Monitor
- Config

Connections:

13 gnutellaNet
3 uploads
1 downloads

gnutellaNet stats:

3 hosts
1,718 files
12,141 MB
18197ms avg ping

[Update](#)

Downloads:

File	Size	Status
DJ Homer - The Simpsons Techno Re...	3356672	57% (15.2k/s)

[Abort selected](#)
[Resume selected](#)
[Clear completed](#)

Maximum simultaneous downloads:

☐ Auto clear completed downloads

Download queue:

File	Size
------	------

[Update](#)
[Remove selected from queue](#)

Copyright (C) 2000, Nullsoft, Inc. - <http://www.gnollsoft.org>

Gnutella v0.48 Mar 13 2000 19:03:46

gnutellaNet

- Uploads
- Downloads
- Search
- Monitor
- Config**

Connections:

14 gnutellaNet
2 uploads
1 downloads

gnutellaNet stats:

3 hosts
1,718 files
12,141 MB
18197ms avg ping

[Update](#)

Save new files to:

D:\Downloads

Path(s) to files

D:\Downloads\Wildrover\mp3\download

[Add directory](#)
Files scanned: 612
[Rescan](#)

Search extensions:

mp3,mp2,mp1,mpg,vqf,avi,mpeg,wav,mod,voc,il,xm,s3m,stm,wma,mov,asf,zip,tar

Listen port

Default: 6346, Disable: 0

[Update](#)

Current port: 22674

Local IP

☐ Force local IP to:

Connection speed

Kbps

☐ Throttle uploads

Searches

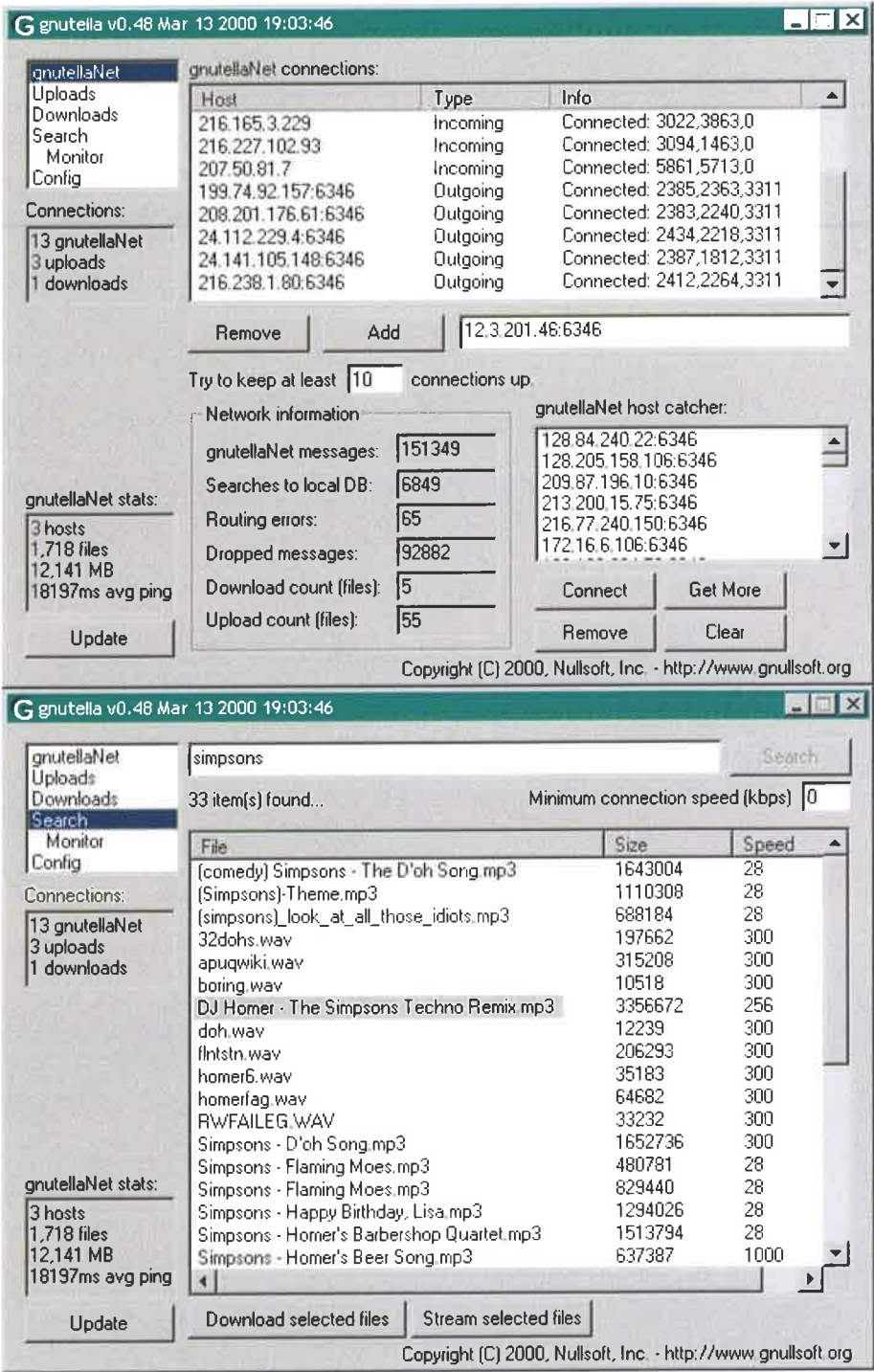
Limit search results to items

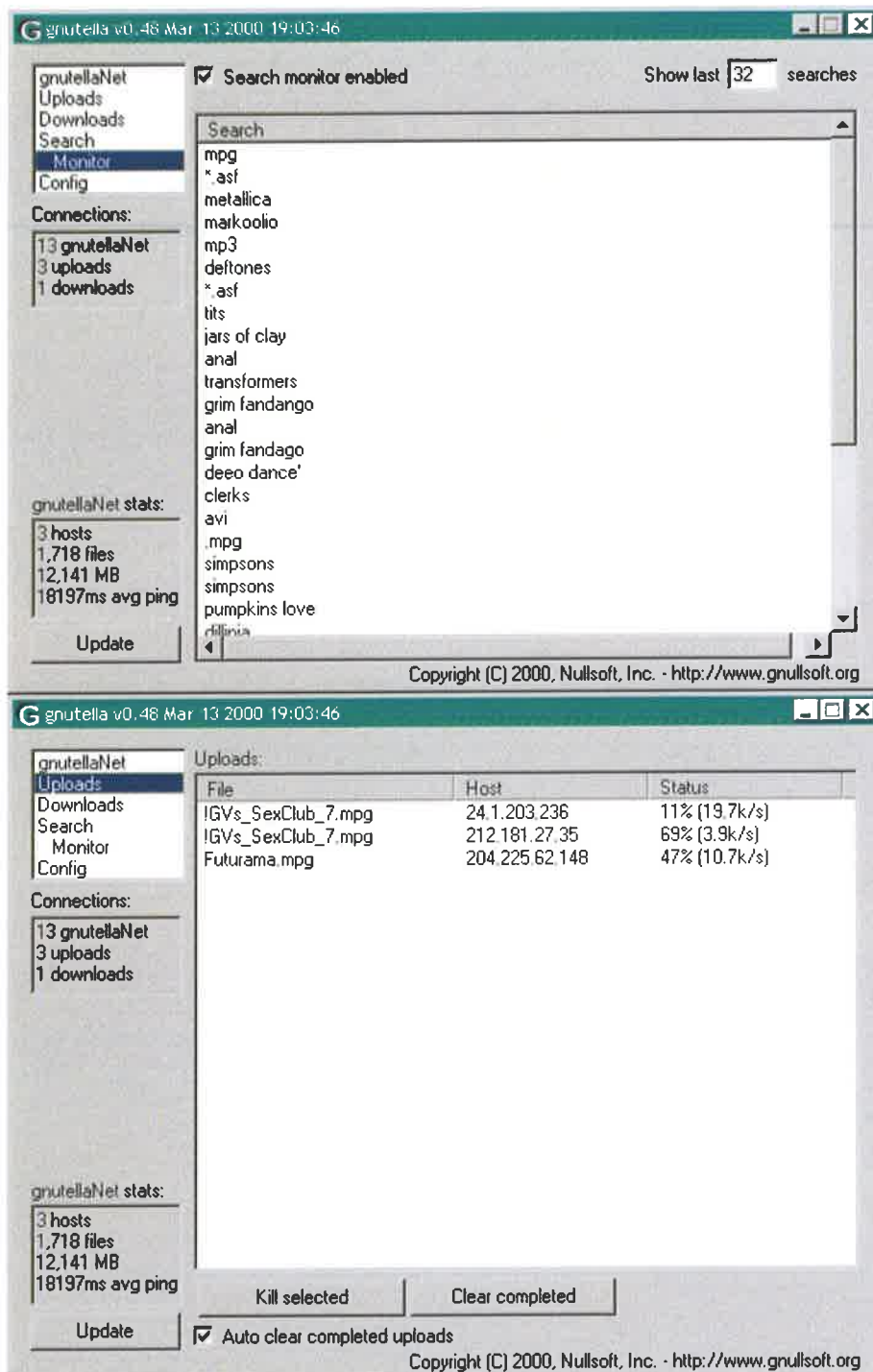
(This limits how many results gnutella returns to other people searching your files)

gnutellaNet TTL settings (both default to 5)

Max TTL: My TTL:

Copyright (C) 2000, Nullsoft, Inc. - <http://www.gnollsoft.org>





[Email CapnBry](#) remove spam from the address for it to work

[Home](#) | [Other things on linux](#)

CapnBry's PHP Gnutella search v0.4 - See source code for licensing information



Cap'n Bry's gnutella search

Search for: Server timeout 60 secs  

- [Problems](#)
- [Documentation?](#)
- [Links](#)
- [The protocol](#)
- [Screenshots](#)
- [My client](#)
- [FAQ's](#)
- [Search](#)

What's it all about?

Way back in the day, when gnutella first went into beta and then was quickly removed from production altogether, I set out to reverse-engineer the protocol so that development could continue on this webvolutionary concept. The fruits of that labor are summarized in my [protocol page](#). Not being one to just live in theory, I then set out to create a client.

I've created a TComponent-derived class in [Delphi 5](#). The concept is that you can drop this component on your form, set some properties and event handlers, and bada-bing you've got fullon gnutellanet access in your application. The source code for this component is located below in the **Downloads** section of this document. This source *requires* Delphi 5 (any edition), as it uses Delphi's standard TServerSocket and TClientSocket and I don't think they had them working right in D4. Some people have requested that I convert the whole thing to use [Jocelle's Internet Component Suite \(ICS\)](#). Maybe I'll do that at some point, but I'm not really familiar with TWSocket and Delphi's sockets have been working well for me so far. Version 2. :)

Downloads

- [gnutellatrans.pas](#) - This is the Delphi (Object Pascal) source code for the component prettied up and put into HTML, just like it looks in the Delphi IDE. Pretty sweet eh?
- [Component and sample app](#) - This ZIP file includes the component source, a pretty icon for your "Internet" component palette, and a sample application which shows how to hook up to component. See the **Known Issues** section below for a list of... well, known issues
- [Screenshots](#) - A quick look at screenshots from the demo app.

Known Issues

- It doesn't know when messages are destined for itself. Need to wrap the sends so that it adds the message to our routing list. Cake walk.
- No implemented buffering system for writing to clients who's winsock buffer is full (i.e. send() returns WSAEWOULDBLOCK). It's coming.
- On occasion, it just stops sending messages to other clients. It does a write on the socket but nothing comes out the other end.
- The "gnutellaNet stats" are just plain screwed up. I haven't even looked at this yet.
- The routing errors and dropped messages counts don't match up with a real gnutella client. I've gotta look closer at what to call what.
- Doesn't do searching of your machine. I'll be implementing this in a plug-in architecture so I haven't worked out the details yet.
- Can't download files. This is done over HTTP so it's a snap to add when I get to it.
- CPU utilization way too high. I haven't done a lick of optimization yet, but I know where the bottlenecks are. I'll probably do this last.

News

June 19, 2000 - New everything uploaded. Known issues fixed:

- Changing your listening port with incoming connections would disconnect the clients, but not fire the event handler, leaving them in not-really-connected limbo. Fixed.
- New Shutdown() function allows closing of all connections so no more events getting

EXHIBIT A

fired while in the destructor!

- Attempting a Listen() of an already in use port no longer throws an exception. Thanks to Andreas Goetz for noticing that this wasn't being caught where I thought it was. (Guess I should actually *try* my code... not just write it)
- Removed logging stream from test app.
- Increased size of routing table to 8k nodes. This is what it was originally, but I lowered it to 128 for a test and forgot to put it back.

June 14, 2000 - Just uploaded a new gnutellatrans.pas and src.zip to the server. Added in this version is message routing (finally) and a merged list of outgoing and incoming clients. Fixes lots of bugs involving "Not a socket" errors when people disconnect.

[Email CapnBry](#) *remove spam from the address for it to work*

[Home](#) | [Other things on linux](#)

CapnBry's PHP Gnutella search v0.4 - See source code for licensing information

```

unit gnutellatrans;

interface

(*****
Copyright (C) 2000, Bryan Mayland Created 2000. All rights reserved.

License to copy and use this software is granted provided that it
is identified as the "CapnBry gnutella transport algorithm"
in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the CapnBry gnutella
transport algorithm" in all material mentioning or referencing the
derived work.

Bryan Mayland makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
*****)

(***)
  Quick overview:
  -- Use Listen() to start accepting incoming connections
  -- Use ConnectNewHost() to initiate an outgoing gnutellanet connection
  -- Use BeginSearch() to ... errr, what does that do again?
  -- Make sure you call Shutdown() before your form's destructor to prevent
      event handlers being fired after your other visual components have been freed
  ***)

uses
  Windows, WinSock, SysUtils, Classes, ScktComp, Messages;

const
  MAX_ROUTE_LIST_SIZE = 8192;

type
  Pgnutella_header = ^gnutella_header;
  gnutella_header = packed record
    MsgID:      TGUID;
    Func:       byte;
    TTLRemaining: byte;
    HopsTaken:   byte;
    DataLen:     integer;
  end;

  Pgnutella_init_response = ^gnutella_init_response;
  gnutella_init_response = packed record
    port:      WORD;
    ip:        Integer;
    filecnt:   DWORD;
    totsize:   DWORD;
  end;

  Pgnutella_query_response_hdr = ^gnutella_query_response_hdr;
  gnutella_query_response_hdr = packed record
    nrecs:     BYTE;
    port:      WORD;
    ip:        Integer;
    speed:     Integer;
  end;

  Pgnutella_query_response_rec_hdr = ^gnutella_query_response_rec_hdr;
  gnutella_query_response_rec_hdr = packed record
    FileIdx:   Integer;
    FileSize:  Integer;
  end;

```

EXHIBIT A

```

Pgnutella_query_response_ftr = ^gnutella_query_response_ftr;
gnutella_query_response_ftr = packed record
  clientid128:   TGUID;
end;

Pgnutella_push_req = ^gnutella_push_req;
gnutella_push_req = packed record
  ClientID128:   TGUID;
  FileIndex:     integer;
  RequestIP:     integer;
  RequestPort:   WORD;
end;

TGConnection = class(TObject)
private
  FSocket:       TCustomWinSocket;
  FRecvBuf:      TMemoryStream;
  FProtocolConnected: boolean;
  FHost:         string;
  FSocketParent: TObject;
  FIncoming:     boolean;
  FMessagesOut:  DWORD;
  FMessagesDropped: DWORD;
  FMessagesIn:   DWORD;
  function GetRecvBufHead: Pgnutella_header;
  function GetRecvBufPos: PChar;
public
  constructor Create(ASockParent : TObject; ASocket : TCustomWinSocket;
    const AHost : string);
  destructor Destroy; override;

  function ReadFromSocket : integer;
  procedure TransmitMsg(SrcGConn : TGConnection);

  property RecvBufHead: Pgnutella_header read GetRecvBufHead;
  property RecvBufPos: PChar read GetRecvBufPos;
  property Host: string read FHost;
  property Incoming: boolean read FIncoming write FIncoming;
  property ProtocolConnected: boolean read FProtocolConnected write FProtocolConnected;
  property RecvBuf: TMemoryStream read FRecvBuf;
  property Socket: TCustomWinSocket read FSocket;
  property MessagesIn: DWORD read FMessagesIn;
  property MessagesOut: DWORD read FMessagesOut;
  property MessagesDropped: DWORD read FMessagesDropped;
end;

TStringEvent = procedure (Sender : TObject; const s : string) of Object;
TClientSockEvent = procedure (GConn : TGConnection) of Object;
TInitRespEvent = procedure (const HostIP : string; HostPort : WORD;
  FileCnt: integer; FileSize: integer) of Object;
TSearchReqEvent = procedure (MinSpeed : WORD; const Search : string) of Object;
TSrchRespBeginEvent = procedure (const HostIP : string; HostPort : WORD;
  Speed : WORD) of Object;
TSrchRespItemEvent = procedure (FileIdx, FileSize, HostSpeed : integer;
  const FileName : string) of Object;
TSrchRespEndEvent = procedure (ClientID128 : TGUID) of Object;

PGnuTelMessage = ^TGnuTelMessage;
TGnuTelMessage = record
  MessageID:   TGUID;
  SourceConn:  TGConnection;
  Next:        PGnuTelMessage;
  Prev:        PGnuTelMessage;
end;

TGnuTelMessageList = class(TObject)
private
  FHead:      PGnuTelMessage;
  FTail:      PGnuTelMessage;
  FCount:     integer;
  function RemoveNode(pMsg : PGnuTelMessage) : PGnuTelMessage;
protected
public
  constructor Create;

```

```

destructor Destroy; override;

procedure Append(GConn : TGConnection);
function FindByMessageID(const Hdr : gnutella_header) : PGnuTelMessage;
procedure PurgeGConn(GConn : TGConnection);

property Count: integer read FCount;
end;

TGNutellaTrans = class(TComponent)
private
  ConnList: TList;
  ServerSock: TServerSocket;
  FGTAvgPing: integer;
  FSearchesIn: DWORD;
  FGTHosts: integer;
  FGTHSize: integer;
  FGTHFiles: integer;
  FMessagesIn: DWORD;
  FOnLogStatus: TStringEvent;
  FOnClientConnected: TClientSockEvent;
  FOnClientCreated: TClientSockEvent;
  FOnClientProtocolConnect: TClientSockEvent;
  FMyTTL: BYTE;
  FMaxTTL: BYTE;
  FOnInitResponse: TInitRespEvent;
  FOnSearchReq: TSearchReqEvent;
  FOnPushRequest: TNotifyEvent;
  FOnSrchRespBegin: TSrchRespBeginEvent;
  FOnSrchRespEnd: TSrchRespEndEvent;
  FOnSrchRespItem: TSrchRespItemEvent;
  FOnClientDisconnect: TClientSockEvent;
  FListenPort: integer;
  FLocalIP: string;
  FLocalFileSize: integer;
  FLocalFileCount: integer;
  FSendInitOnConnect: boolean;
  FOnCatchHost: TStringEvent;
  FConnectionSpeed: integer;
  FRouteMsgList: TGNUtelMessageList;
  FOnListenPortInUse: TNotifyEvent;
  FRoutingErrorsIn: DWORD;
  FGConnSelf: TGConnection;
  FMessagesDropped: DWORD;
  FFilesUploaded: DWORD;
  FFilesDownloaded: DWORD;

  procedure GetGNUTELLAOK(Socket: TCustomWinSocket);
  procedure GetGNUTELLA_CONNECT(Socket: TCustomWinSocket);
  procedure ReadSocketData(Socket: TCustomWinSocket);
  procedure DealWithData(GConn : TGConnection; pData : PChar; DataSize : integer);
  procedure GotMessage(GConn : TGConnection);
  procedure hdr_to_text(const hdr: gnutella_header);
  function FuncToStr(Func: BYTE): string;
  procedure getInit(GConn : TGConnection);
  procedure getInitResponse(GConn : TGConnection);
  procedure getQuery(GConn : TGConnection);
  procedure getQueryResponse(GConn : TGConnection);
  procedure getUnknown(GConn : TGConnection);
  procedure getPushReq(GConn : TGConnection);
  function GetConnectionCount: integer;
  function GetSocketCount: integer;
  procedure SetListenPort(const Value: integer);
  procedure SendMessageToAll(pBuff: Pointer; Size: integer);
  function RouteMessageResponse(GConn : TGConnection) : boolean;
  function BroadcastMessage(GConn : TGConnection) : boolean;
  procedure HandleDisconnError(const sMsg: string; GConn: TGConnection;
    var ErrorCode: integer);
  function HeaderValid(const hdr: gnutella_header) : boolean;
protected
  procedure Log(const s : string);

  procedure sockConnect(Sender: TObject; Socket: TCustomWinSocket);
  procedure sockRead(Sender: TObject; Socket: TCustomWinSocket);

```

EXHIBIT A

```

procedure sockDisconnect(Sender: TObject; Socket: TCustomWinSocket);
procedure sockError(Sender: TObject; Socket: TCustomWinSocket;
  ErrorEvent: TErrorEvent; var ErrorCode: Integer);
procedure sockAccept(Sender: TObject; Socket: TCustomWinSocket);
procedure SetupNewHdr(var hdr : gnutella_header);
procedure FreeAllSockets;
public
  constructor Create(AOwner : TComponent); override;
  destructor Destroy; override;

  procedure ConnectNewHost(const Host : string);
  procedure ClearGTNStats;
  procedure SendInitToAll;
  procedure SendOneInit(Socket : TCustomWinSocket);
  procedure BeginSearch(const Query : string; MinSpeed : WORD);
  procedure Listen;
  procedure CancelListen;
  procedure Shutdown;
  procedure DisconnectHost(const Host : string);

  property MessagesIn: DWORD read FMessagesIn;
  property SearchesIn: DWORD read FSearchesIn;
  property RoutingErrorsIn: DWORD read FRoutingErrorsIn;
  property MessagesDropped: DWORD read FMessagesDropped;
  property FilesUploaded: DWORD read FFilesUploaded;
  property FilesDownloaded: DWORD read FFilesDownloaded;
  property GTNHosts: integer read FGTNHosts;
  property GTNFiles: integer read FGTNFiles;
  property GTNSize: integer read FGTNSize;
  property GTNAvgPing: integer read FGTNAvgPing;
  property ConnectionCount: integer read GetConnectionCount;
  property SocketCount: integer read GetSocketCount;
  property LocalFileCount: integer read FLocalFileCount;
  property LocalFileSize: integer read FLocalFileSize;
published
  property ConnectionSpeed: integer read FConnectionSpeed write FConnectionSpeed;
  property ListenPort: integer read FListenPort write SetListenPort;
  property LocalIP: string read FLocalIP write FLocalIP;
  property MyTTL: BYTE read FMyTTL write FMyTTL;
  property MaxTTL: BYTE read FMaxTTL write FMaxTTL;
  property SendInitOnConnect: boolean read FSendInitOnConnect write FSendInitOnConnect;
  property OnCatchHost: TStringEvent read FOnCatchHost write FOnCatchHost;
  property OnClientConnected: TClientSockEvent read FOnClientConnected
    write FOnClientConnected;
  property OnClientCreated: TClientSockEvent read FOnClientCreated
    write FOnClientCreated;
  property OnClientProtocolConnect: TClientSockEvent read FOnClientProtocolConnect
    write FOnClientProtocolConnect;
  property OnInitResponse: TInitRespEvent read FOnInitResponse write FOnInitResponse;
  property OnListenPortInUse: TNotifyEvent read FOnListenPortInUse write FOnListenPortInUse;
  property OnLogStatus: TStringEvent read FOnLogStatus write FOnLogStatus;
  property OnSearchReq: TSearchReqEvent read FOnSearchReq write FOnSearchReq;
  property OnSrchRespBegin: TSrchRespBeginEvent read FOnSrchRespBegin
    write FOnSrchRespBegin;
  property OnSrchRespEnd: TSrchRespEndEvent read FOnSrchRespEnd
    write FOnSrchRespEnd;
  property OnSrchRespItem: TSrchRespItemEvent read FOnSrchRespItem
    write FOnSrchRespItem;
  property OnPushRequest: TNotifyEvent read FOnPushRequest write FOnPushRequest;
  property OnClientDisconnect: TClientSockEvent read FOnClientDisconnect
    write FOnClientDisconnect;
end;

function CoCreateGuid(out guid: TGUID): HRESULT; stdcall; external 'ole32.dll';
function StringFromCLSID(const clsid: TGUID; out psz: PWideChar): HRESULT;
  stdcall; external 'ole32.dll';
procedure CoTaskMemFree(pv: Pointer); stdcall; external 'ole32.dll';

procedure Register;

implementation

{$R *.dcr}

```


EXHIBIT A

```

procedure Register;
begin
  RegisterComponents('Internet', [TGnutellaTrans]);
end;

function GUIDToString(const ClassID: TGUID): string;
var
  P: PWideChar;
begin
  StringFromCLSID(ClassID, P);
  Result := P;
  CoTaskMemFree(P);
end;

function PointerToStr(P : Pointer; iSize : integer) : string;
begin
  Result := '';
  while iSize > 0 do begin
    if iSize > 1 then
      AppendStr(Result, Format('%2.2x ', [PBYTE(P)^]))
    else
      AppendStr(Result, Format('%2.2x', [PBYTE(P)^]));
    dec(iSize);
    Inc(PBYTE(P), 1)
  end;
end;

function BytesToStr(ar : array of byte) : string;
var
  I: integer;
begin
  Result := '';
  for I := low(ar) to high(ar) do
    if I < high(ar) then
      AppendStr(Result, Format('%2.2x ', [ar[i]]))
    else
      AppendStr(Result, Format('%2.2x', [ar[i]]))
  end;

{ TGnutellaTrans }

function TGnutellaTrans.FuncToStr(Func : BYTE) : string;
begin
  case Func of
    $00: Result := 'Init';
    $01: Result := 'Init response';
    $40: Result := 'Client push req';
    $80: Result := 'Query';
    $81: Result := 'Query Response';
  else
    Result := '???';
  end;
end;

procedure TGnutellaTrans.ClearGTNStats;
begin
  FGTNAvgPing := 0;
  FGTNHosts := 0;
  FGTNSize := 0;
  FGTNFiles := 0;
end;

procedure TGnutellaTrans.ConnectNewHost(const Host: string);
var
  tmpSocket: TClientSocket;
  GConn: TGConnection;
  iPos: integer;
begin
  if Host = '' then
    exit;

  tmpSocket := TClientSocket.Create(nil);

```


EXHIBIT A

```

GConn := TGConnection.Create(tmpSocket, tmpSocket.Socket, Host);
ConnList.Add(GConn);

iPos := Pos(':', Host);
if iPos = 0 then
    iPos := Length(Host) + 1;

tmpSocket.Host := copy(Host, 1, iPos - 1);
tmpSocket.Port := StrToIntDef(copy(Host, iPos + 1, Length(Host)), 6346);

Log('Host is: ' + tmpSocket.Host);
Log('Port is: ' + IntToStr(tmpSocket.Port));

with tmpSocket do begin
    OnConnect := sockConnect;
    OnDisconnect := sockDisconnect;
    OnError := sockError;
    OnRead := sockRead;
    Open;
end;

if Assigned(FOnClientCreated) then
    FOnClientCreated(GConn);
end;

constructor TGnutellaTrans.Create(AOwner: TComponent);
begin
    inherited;

    FRouteMsgList := TGnuTelMessageList.Create;
    { we create a bogus connection to represent ourself }
    FGConnSelf := TGConnection.Create(nil, nil, 'Self');

    ConnList := TList.Create;
    FMyTTL := 7;
    FMaxTTL := 7;
    FListenPort := 6346;
    FConnectionSpeed := 28;

    ServerSock := TServerSocket.Create(Self);
    ServerSock.Port := FListenPort;
    ServerSock.OnClientConnect := sockConnect;
    ServerSock.OnClientDisconnect := sockDisconnect;
    ServerSock.OnClientError := sockError;
    ServerSock.OnClientRead := sockRead;
    ServerSock.OnAccept := sockAccept;
end;

destructor TGnutellaTrans.Destroy;
begin
    Shutdown;
    ConnList.Free;
    FRouteMsgList.Free;
    FGConnSelf.Free;

    inherited;
end;

procedure TGnutellaTrans.GetGNUTELLAOK(Socket: TCustomWinSocket);
var
    GConn: TGConnection;
    iBytes: integer;
begin
    GConn := TGConnection(Socket.Data);
    { 13 is for GNUTELLA OKaa }
    GConn.FRecvBuf.Size := 13;

    iBytes := GConn.ReadFromSocket;
    if iBytes = SOCKET_ERROR then begin
        Log('Socket error in GNUTELLA OK');
        Socket.Close;
    end
    else begin
        GConn.RecvBuf.Position := GConn.RecvBuf.Position + iBytes;
    
```

EXHIBIT A

```

    if GConn.RecvBuf.Position = GConn.RecvBuf.Size then
        if StrLComp(PChar('GNUTELLA OK'#10#10), PChar(GConn.RecvBufHead),
            GConn.RecvBuf.Size) = 0 then begin
            GConn.RecvBuf.Clear;
            GConn.ProtocolConnected := true;
            if Assigned(FOnClientProtocolConnect) then
                FOnClientProtocolConnect(GConn);
            if FSendInitOnConnect then
                SendOneInit(Socket);
        end // we got GNUTELLA OK

        else begin
            Log('Got invalid GNUTELLA response');
            Socket.Close;
        end; // not GNUTELLA OK
    end; // if no SOCKET_ERROR
end;

procedure TGnutellaTrans.Log(const s: string);
begin
    if Assigned(FOnLogStatus) then
        FOnLogStatus(Self, s);
end;

procedure TGnutellaTrans.ReadSocketData(Socket: TCustomWinSocket);
var
    Buf: array[0..8195] of char;
    GConn: TGConnection;
    iBytes: integer;
begin
    GConn := TGConnection(Socket.Data);

    repeat
        iBytes := Socket.ReceiveBuf(Buf, sizeof(Buf));

        if iBytes <> SOCKET_ERROR then
            DealWithData(GConn, @Buf, iBytes);
    until iBytes < sizeof(Buf);
end;

procedure TGnutellaTrans.SendMessageToAll(pBuff : Pointer; Size : integer);
var
    I: integer;
begin
    for I := 0 to ConnList.Count - 1 do
        with TGConnection(ConnList[I]) do
            if ProtocolConnected then
                Socket.SendBuf(pBuff^, Size);
    end;

procedure TGnutellaTrans.SendInitToAll;
var
    hdr: gnutella_header;
begin
    SetupNewHdr(hdr);
    hdr.Func := 0;
    SendMessageToAll(@hdr, sizeof(hdr));
end;

procedure TGnutellaTrans.SetupNewHdr(var hdr: gnutella_header);
begin
    FillChar(hdr, sizeof(hdr), 0);
    CoCreateGUID(hdr.MsgID);
    hdr.TTLRemaining := FMyTTL;
end;

procedure TGnutellaTrans.sockConnect(Sender: TObject;
    Socket: TCustomWinSocket);
var
    GConn: TGConnection;
begin
    { if socket.data is not assigned, then this is an incoming connection }
    if not Assigned(Socket.Data) then begin
        GConn := TGConnection.Create(nil, Socket,

```

EXHIBIT A

```

    Format('%s:%d', [Socket.RemoteAddress, Socket.LocalPort]);
    GConn.Incoming := true;
    ConnList.Add(GConn);
    if Assigned(FOnClientCreated) then
        FOnClientCreated(GConn);
end;

    { if the socket is outbound, we need to init the protocol }
if not TGConnection(Socket.Data).Incoming then
    Socket.SendText('GNUTELLA CONNECT/0.4'#10#10);

Log('SocketConnect');

if Assigned(FOnClientConnected) then
    FOnClientConnected(TGConnection(Socket.Data));
end;

procedure TGnutellaTrans.sockDisconnect(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    Log('socket disconnected');
    { here's a super kludge }
    PostMessage(Socket.Handle, CM_SOCKETMESSAGE, Socket.SocketHandle,
        MakeLParam(FD_CLOSE, WSAECONNRESET));
end;

procedure TGnutellaTrans.HandleDisconnError(const sMsg : string; GConn : TGConnection;
    var ErrorCode : integer);
var
    I: integer;
begin
    Log(sMsg);
    if Assigned(GConn) then begin
        { we delete the socket before we call the disconnect handler, so
            the user has an accurate disconnected client count in the handler }
        I := ConnList.IndexOf(GConn);
        if I <> -1 then
            ConnList.Delete(I);

        { remove messages destined for this guy from the RouteList }
        FRouteMsgList.PurgeGConn(GConn);

        if Assigned(FOnClientDisconnect) then
            FOnClientDisconnect(GConn);

        GConn.Free;
    end;

    ErrorCode := 0;
end;

procedure TGnutellaTrans.sockError(Sender: TObject;
    Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
    var ErrorCode: Integer);
begin
    if ErrorCode = WSAEADDRINUSE then begin // 10048
        Log('Listen port in use');
        if Assigned(FOnListenPortInUse) then
            FOnListenPortInUse(Self);
        ErrorCode := 0;
    end

    else case ErrorCode of
        WSAEADDRNOTAVAIL: // 10049
            HandleDisconnError('Host unreachable', TGConnection(Socket.Data), ErrorCode);
        WSAENETUNREACH: // 10051
            HandleDisconnError('Network unreachable', TGConnection(Socket.Data), ErrorCode);
        WSAECONNABORTED: // 10053
            HandleDisconnError('Connection aborted', TGConnection(Socket.Data), ErrorCode);
        WSAECONNRESET: // 10054
            HandleDisconnError('Connection reset', TGConnection(Socket.Data), ErrorCode);
        WSAETIMEDOUT: // 10060
            HandleDisconnError('Connection timed out', TGConnection(Socket.Data), ErrorCode);
        WSAECONNREFUSED: // 10061

```

EXHIBIT A

```

    HandleDisconnError('Connection refused', TGConnection(Socket.Data), ErrorCode);
WSAEHOSTUNREACH:    // 10065
    HandleDisconnError('Host unreachable', TGConnection(Socket.Data), ErrorCode);
else
    HandleDisconnError('Socket Error ' + IntToStr(ErrorCode),
        TGConnection(Socket.Data), ErrorCode);
end; { case ErrorCode }
end;

procedure TGNutellaTrans.sockRead(Sender: TObject;
    Socket: TCustomWinSocket);
begin
    { if we haven't set our own IP, this is the time to do it }
    if FLocalIP = '' then
        if Socket.LocalAddress <> '127.0.0.1' then
            FLocalIP := Socket.LocalAddress;

    if not TGConnection(Socket.Data).ProtocolConnected then
        if TGConnection(Socket.Data).Incoming then
            GetGNUTELLA_CONNECT(Socket)
        else
            GetGNUTELLAOK(Socket)
    else
        ReadSocketData(Socket);
end;

procedure TGNutellaTrans.SendOneInit(Socket: TCustomWinSocket);
var
    hdr:    gnutella_header;
begin
    SetupNewHdr(hdr);
    hdr.Func := 0;
    Socket.SendBuf(hdr, sizeof(hdr));
end;

procedure TGNutellaTrans.DealWithData(GConn: TGConnection; pData: PChar;
    DataSize: integer);
var
    iBytesNeeded: integer;
    iBytesDealt: integer;
begin
    iBytesDealt := 0;
    while iBytesDealt < DataSize do begin
        { check for an previous underflow }
        iBytesNeeded := sizeof(gnutella_header) + GConn.RecvBuf.Position;
        if iBytesNeeded > (DataSize - iBytesDealt) then
            iBytesNeeded := DataSize - iBytesDealt;

        GConn.RecvBuf.Write(PChar(Integer(pData) + iBytesDealt)^, iBytesNeeded);
        inc(iBytesDealt, iBytesNeeded);

        { we don't have enough data for to be a header }
        if GConn.RecvBuf.Position < sizeof(gnutella_header) then
            break;

        iBytesNeeded := GConn.RecvBufHead^.DataLen;

        if not HeaderValid(GConn.RecvBufHead^) then begin
            Log('I think I've got a bogus packet:');
            hdr_to_text(GConn.RecvBufHead^);
            GConn.RecvBuf.Clear;
            continue;
        end;

        { check for an underflow }
        if iBytesNeeded > DataSize - iBytesDealt then
            iBytesNeeded := DataSize - iBytesDealt;
        GConn.RecvBuf.Write(PChar(Integer(pData) + iBytesDealt)^, iBytesNeeded);
        inc(iBytesDealt, iBytesNeeded);

        { we don't have the whole message yet }
        if GConn.RecvBuf.Position <
            (sizeof(gnutella_header) + (GConn.RecvBufHead^.DataLen)) then
            break;
    end;
end;

```

EXHIBIT A

```

    GotMessage(GConn);
    GConn.RecvBuf.Clear;
end; { while Bytes }
end;

procedure TGNutellaTrans.hdr_to_text(const hdr: gnutella_header);
begin
{ BRY!
if hdr.Func in [$00, $01, $40, $80, $81] then
exit;
}
    Log('Message ID: ' + GUIDToString(hdr.MsgID));
    Log('Function ID: 0x' + Format('%2.2x', [hdr.Func]) + ' (' + FuncToStr(hdr.Func) + ')');
    Log('TTL Remaining: ' + IntToStr(hdr.TTLRemaining));
    Log('Hops Taken: ' + IntToStr(hdr.HopsTaken));
    Log('Data Len: ' + IntToStr(hdr.DataLen));
end;

procedure TGNutellaTrans.GotMessage(GConn: TGConnection);
begin
    hdr_to_text(GConn.RecvBufHead^);

    inc(GConn.FMessagesIn);

    if GConn.RecvBufHead^.Func and $01 = $01 then
        { if route... returns true, the message is for us! }
        if RouteMessageResponse(GConn) then begin
            case GConn.RecvBufHead^.Func of
                $01: getInitResponse(GConn);
                $81: getQueryResponse(GConn);
            else
                getUnknown(GConn);
            end;
        end { if response message for us }

    else begin
        end { message not for us }

    else begin
        { if broadcast returns true, then we should respond }
        if BroadcastMessage(GConn) then begin
            case GConn.RecvBufHead^.Func of
                $00: getInit(GConn);
                $40: getPushReq(GConn);
                $80: getQuery(GConn);
            else
                getUnknown(GConn);
            end;
        end; { if new message }
    end;

    inc(FMessagesIn);
end;

procedure TGNutellaTrans.getInit(GConn: TGConnection);
type
    gir = packed record
        hdr: gnutella_header;
        ir: gnutella_init_response;
    end;
var
    gir_out: gir;
begin
    Log('Init message in, sending response...');
    fillchar(gir_out, sizeof(gir_out), 0);
    move(GConn.RecvBufHead^, gir_out.hdr, sizeof(gnutella_header));
    gir_out.hdr.Func := 1;
    gir_out.hdr.DataLen := sizeof(gir_out.ir);
    gir_out.ir.Port := htons(FListenPort);
    gir_out.ir.ip := inet_addr(PChar(FLocalIP));
    gir_out.ir.filecnt := FLocalFileCount;
    gir_out.ir.totsize := FLocalFileSize;
    GConn.Socket.SendBuf(gir_out, sizeof(gir_out));

```

EXHIBIT A

```

end;

procedure TGNutellaTrans.getInitResponse(GConn: TGConnection);
var
  pir:   Pgnutella_init_response;
  ia:    in_addr;
  sAddr: string;
begin
  Log('Init response:');

  pir := Pgnutella_init_response(
    integer(GConn.RecvBufHead) + sizeof(gnutella_header));

  inc(FGTNHosts);
  inc(FGTNFiles, pir^.FileCnt);
  { totdsize is in KB, we want MB }
  inc(FGTNSize, pir^.totdsize div 1024);

  Log(' Remote Port: ' + IntToStr(pir^.Port));
  ia.S_addr := pir^.ip;
  sAddr := inet_ntoa(ia);
  Log(' Remote IP: ' + sAddr);
  Log(' Files shared: ' + IntToStr(pir^.filecnt));
  Log(' Total Size: ' + IntToStr(pir^.totdsize) + ' KB');

  if Assigned(FOnCatchHost) then
    FOnCatchHost(Self, Format('%s:%d', [sAddr, pir^.Port]));

  if Assigned(FOnInitResponse) then
    FOnInitResponse(sAddr, pir^.Port, pir^.filecnt, pir^.totdsize);
end;

procedure TGNutellaTrans.getPushReq(GConn: TGConnection);
var
  ppr:   Pgnutella_push_req;
  sAddr: string;
  ia:    in_addr;
begin
  Log('Client Push request:');

  ppr := Pgnutella_push_req(
    Integer(GConn.RecvBufHead) + sizeof(gnutella_header));

  Log(' ClientID128: ' + GUIDToString(ppr^.ClientID128));
  Log(' FileIndex: ' + IntToStr(ppr^.FileIndex));
  ia.S_addr := ppr^.RequestIP;
  sAddr := string(inet_ntoa(ia));
  Log(' Requester IP: ' + sAddr);
  Log(' Requester Port: ' + IntToStr(ppr^.RequestPort));

  if Assigned(FOnCatchHost) then
    FOnCatchHost(Self, Format('%s:%d', [sAddr, ppr^.RequestPort]));

  if Assigned(FOnPushRequest) then
    FOnPushRequest(Self);
end;

procedure TGNutellaTrans.getQuery(GConn: TGConnection);
var
  wMinSpeed: WORD;
  sSearch:   string;
begin
  wMinSpeed := PWORD(Integer(GConn.RecvBufHead) + sizeof(gnutella_header));
  sSearch := string(PChar(
    Integer(GConn.RecvBufHead) + sizeof(gnutella_header) + 2));

  inc(FSearchesIn);

  Log('Query:');
  Log('Minimum Speed: ' + IntToStr(wMinSpeed));
  Log('Search Criteria: ' + sSearch);

  if Assigned(FOnSearchReq) then
    FOnSearchReq(wMinSpeed, sSearch);

```



```

end;

procedure TGNutellaTrans.getQueryResponse(GConn: TGConnection);
var
  pqrh: Pgnutella_query_response_hdr;
  ia: in_addr;
  P: Pointer;
  PinP: PChar;
  I: integer;
  s: string;
  iFileSize: integer;
  iFileIdx: integer;
begin
  Log('Query Response:');
  pqrh := Pgnutella_query_response_hdr(
    integer(GConn.RecvBufHead) + sizeof(gnutella_header));

  Log(' Remote Port: ' + IntToStr(pqrh^.port));
  ia.S_addr := pqrh^.ip;
  s := inet_ntoa(ia);
  Log(' Remote IP: ' + s);
  Log(' Remote Speed: ' + IntToStr(pqrh^.speed));

  if Assigned(FOnCatchHost) then
    FOnCatchHost(Self, Format('%s:%d', [s, pqrh^.port]));

  if Assigned(FOnSrchRespBegin) then
    FOnSrchRespBegin(s, pqrh^.port, pqrh^.speed);

  P := Pointer(integer(GConn.RecvBufHead) + sizeof(gnutella_header)
    + sizeof(gnutella_query_response_hdr));
  PinP := P;
  for I := 1 to pqrh^.nrecs do begin
    iFileIdx := Pgnutella_query_response_rec_hdr(PinP)^.FileIdx;
    Log(' File Index: ' + IntToStr(iFileIdx));
    iFileSize := Pgnutella_query_response_rec_hdr(PinP)^.FileSize;
    Log(' File Size: ' + IntToStr(iFileSize));
    inc(PinP, sizeof(gnutella_query_response_rec_hdr));
    s := string(PChar(PinP));
    Log(' File Name: ' + s);

    inc(PinP, Length(s) + 2);
    if Assigned(FOnSrchRespItem) then
      FOnSrchRespItem(iFileIdx, iFileSize, pqrh^.speed, s);
  end; { for Recs }

  Log(' ClientID128: ' + GUIDToString(
    Pgnutella_query_response_ftr(PinP)^.clientid128
  ));

  if Assigned(FOnSrchRespEnd) then
    FOnSrchRespEnd(Pgnutella_query_response_ftr(PinP)^.clientid128);
end;

procedure TGNutellaTrans.getUnknown(GConn: TGConnection);
begin
  Log('Unknown Data: ');

  Log(#13#10 + PointerToStr(
    PChar(Integer(GConn.RecvBufHead) + sizeof(gnutella_header)),
    GConn.RecvBufHead^.DataLen));
end;

procedure TGNutellaTrans.BeginSearch(const Query: string; MinSpeed: WORD);
var
  hdr: gnutella_header;
  P: PChar;
  I: integer;
begin
  SetupNewHdr(hdr);
  hdr.Func := $80;
  hdr.DataLen := 2 + Length(Query) + 1;
  GetMem(P, sizeof(hdr) + 2 + Length(Query) + 1);
  try

```

EXHIBIT A

```

    FillChar(P^, sizeof(Hdr) + 2 + Length(Query) + 1, 0);
    Move(hdr, P^, sizeof(hdr));
    Move(MinSpeed, Pointer(Integer(P) + sizeof(Hdr))^, sizeof(MinSpeed));
    StrPCopy(PChar(Integer(P) + sizeof(Hdr) + 2), PChar(Query));
    for I := 0 to ConnList.Count - 1 do
        if TGConnection(ConnList[I]).ProtocolConnected then
            TGConnection(ConnList[I]).Socket.SendBuf(P^, sizeof(Hdr) + 2 + Length(Query) + 1);
    finally
        FreeMem(P);
    end;
end;

function TGnutellaTrans.GetConnectionCount: integer;
var
    I: integer;
begin
    Result := 0;
    for I := 0 to ConnList.Count - 1 do
        if TGConnection(ConnList[I]).ProtocolConnected then
            inc(Result);
    end;

function TGnutellaTrans.GetSocketCount: integer;
begin
    Result := ConnList.Count;
end;

procedure TGnutellaTrans.FreeAllSockets;
var
    I: integer;
begin
    for I := 0 to ConnList.Count - 1 do
        TGConnection(ConnList[I]).Free;

    ConnList.Clear;
end;

procedure TGnutellaTrans.SetListenPort(const Value: integer);
begin
    if FListenPort = Value then
        exit;

    if ServerSock.Active then
        raise Exception.Create('Setting port while server open not yet supported');

    FListenPort := Value;
    ServerSock.Port := FListenPort;
end;

procedure TGnutellaTrans.GetGNUTELLA_CONNECT(Socket: TCustomWinSocket);
var
    GConn: TGConnection;
    iBytes: integer;
begin
    GConn := TGConnection(Socket.Data);
    { 22 is for GNUTELLA CONNECT/0.4aa }
    GConn.RecvBuf.Size := 22;

    iBytes := GConn.ReadFromSocket;
    if iBytes = SOCKET_ERROR then begin
        Log('Socket error in GNUTELLA CONNECT/0.4');
        Socket.Close;
    end
    else begin
        GConn.RecvBuf.Position := GConn.RecvBuf.Position + iBytes;
        if GConn.RecvBuf.Position = GConn.RecvBuf.Size then
            if StrLComp(PChar('GNUTELLA CONNECT/0.4'#10#10), PChar(GConn.RecvBufHead),
                GConn.RecvBuf.Size) = 0 then begin
                GConn.RecvBuf.Clear;
                Socket.SendText('GNUTELLA OK'#10#10);
                GConn.ProtocolConnected := true;
                if Assigned(FOnClientProtocolConnect) then
                    FOnClientProtocolConnect(GConn);
                if FSendInitOnConnect then

```

EXHIBIT A

```

        SendOneInit(Socket);
    end // we got GNUTELLA OK

    else begin
        Log('Got invalid GNUTELLA protocol init');
        Socket.Close;
    end; // not GNUTELLA OK
end; // if no SOCKET_ERROR
end;

procedure TGNutellaTrans.CancelListen;
var
    I: integer;
    Err: Integer;
begin
    for I := 0 to ServerSock.Socket.ActiveConnections - 1 do begin
        { we get the disconnect message, but since we do a postmessage, we'll
          never get the WSAECONNRESET, so we have to call directly to make
          sure we free our stuff }
        Err := WSAECONNRESET;
        HandleDisconnError('Canceled Listen',
            TGConnection(ServerSock.Socket.Connections[I].Data), Err);
        ServerSock.Socket.Connections[I].Close;
    end;

    ServerSock.Close;
end;

procedure TGNutellaTrans.Listen;
begin
    try
        ServerSock.Open;
    except
        on E : ESocketError do begin
            { this is most likely an Address in Use error }
            // if E.ErrorCode = WSAEADDRINUSE then begin // 10048
            Log('Listen port in use');
            if Assigned(FOnListenPortInUse) then
                FOnListenPortInUse(Self);
        end;
    end;
end;

procedure TGNutellaTrans.sockAccept(Sender: TObject;
    Socket: TCustomWinSocket);
begin
end;

(**
    Returns true if the message is new
**)
function TGNutellaTrans.BroadcastMessage(GConn: TGConnection) : boolean;
var
    pMsg: PGnuTelMessage;
    I: integer;
begin
    pMsg := FRouteMsgList.FindByMessageID(GConn.RecvBufHead^);
    { if we've already seen it, eat it }
    if Assigned(pMsg) then begin
        Result := false;
        log('Duplicate message');
        inc(FMessagesDropped);
        inc(GConn.FMessagesDropped);
    end
    else begin
        Result := true;
        FRouteMsgList.Append(GConn);

        { if we've got some ttl left on it, pass it on }
        if GConn.RecvBufHead^.TTLRemaining > 1 then begin
            GConn.RecvBufHead^.TTLRemaining := GConn.RecvBufHead^.TTLRemaining - 1;

            if GConn.RecvBufHead^.TTLRemaining > FMaxTTL then

```

```

    GConn.RecvBufHead^.TTLRemaining := MaxTTL;

    for I := 0 to ConnList.Count - 1 do
        if TGConnection(ConnList[I]) <> GConn then
            TGConnection(ConnList[I]).TransmitMsg(GConn);
        end;
    end; { if new message }
end;

(**
Returns true if the message is bound for us
**)
function TGNutellaTrans.RouteMessageResponse(GConn: TGConnection) : boolean;
var
    pMsg: PGnuTelMessage;
begin
    Result := false;
    pMsg := FRouteMsgList.FindByMessageID(GConn.RecvBufHead^);
    { if it's in the list, we've already seen it, so send it to the
      owner }
    if Assigned(pMsg) then
        if pMsg^.SourceConn = FGConnSelf then
            Result := true
        else
            pMsg^.SourceConn.TransmitMsg(GConn)
        else
            { we don't know what to do with it! Screw it }
            inc(FRoutingErrorsIn);
        end;
    end;

procedure TGNutellaTrans.DisconnectHost(const Host: string);
var
    I: integer;
begin
    { todo: don't do this by host:port! do it by connection id or something }
    for I := 0 to ConnList.Count - 1 do
        if TGConnection(ConnList[I]).Host = Host then begin
            TGConnection(ConnList[I]).Socket.Close;
            exit;
        end;
    end;

function TGNutellaTrans.HeaderValid(const hdr: gnutella_header): boolean;
begin
    with hdr do
        Result :=
            ((Func = $00) and (DataLen = 0)) or // Pings always have 0 payload
            ((Func = $01) and (DataLen = 14)) or // Pong = 14
            ((Func = $40) and (DataLen = 26)) or // Push req = 26
            (
                ((Func = $80) or (Func = $81)) and
                ((DataLen > 0) and (DataLen < 5000))
            );
    end;

procedure TGNutellaTrans.Shutdown;
begin
    CancelListen;
    FreeAllSockets;
end;

{ TGnuTelMessageList }

procedure TGnuTelMessageList.Append(GConn : TGConnection);
var
    pTmp: PGnuTelMessage;
begin
    New(pTmp);
    Move(GConn.RecvBufHead^.MsgID, pTmp^.MessageID, sizeof(TGUID));
    pTmp^.SourceConn := GConn;

    if Assigned(FHead) then begin
        { new messages go on the front }
        FHead^.Prev := pTmp;
    end;
end;

```

EXHIBIT A

```

pTmp^.Next := FHead;
pTmp^.Prev := nil;
FHead := pTmp;

    { check to see if our list is getting to big }
if FCount < MAX_ROUTE_LIST_SIZE then
    inc(FCount)
else begin
    pTmp := FTail;
    FTail := FTail^.Prev;
    FTail^.Next := nil;
    Dispose(pTmp);
end;
end { if we've got a head }
else begin
    FHead := pTmp;
    FTail := pTmp;
    pTmp^.Next := nil;
    pTmp^.Prev := nil;
    FCount := 1;
end;
end;

constructor TGNUtelMessageList.Create;
begin
    inherited;
    FHead := nil;
    FTail := nil;
    FCount := 0;
end;

destructor TGNUtelMessageList.Destroy;
begin
    while Assigned(FHead) do begin
        { i'm cheating and using FTail as a temp var here }
        FTail := FHead;
        FHead := FHead^.Next;
        Dispose(FTail);
    end;

    inherited;
end;

function TGNUtelMessageList.FindByMessageID(const Hdr : gnutella_header) : PGnuTelMessage;
begin
    { search from the most recent messages to the least recent }
    Result := FHead;

    while Assigned(Result) do
        if CompareMem(@Hdr.MsgID, @Result^.MessageID, sizeof(TGUID)) then
            break
        else
            Result := Result^.Next;
    end;

procedure TGNUtelMessageList.PurgeGConn(GConn: TGConnection);
var
    pTmp: PGnuTelMessage;
begin
    pTmp := FHead;
    while Assigned(pTmp) do begin
        if pTmp^.SourceConn = GConn then
            pTmp := RemoveNode(pTmp);

        pTmp := pTmp^.Next;
    end; { while msgs in the list }
end;

(***)
Removes the Msg node from the list, disposes it, and returns
the Next msg in the list
(***)
function TGNUtelMessageList.RemoveNode(pMsg: PGnuTelMessage) : PGnuTelMessage;
begin

```

EXHIBIT A

```

if pMsg = FHead then begin
  FHead := pMsg^.Next;
  Result := FHead;
  if Assigned(FHead) then
    FHead^.Prev := nil
  else
    { when the head goes nil, the tail must be too }
    FTail := nil;
end { if head }

else if pMsg = FTail then begin
  { we don't have to do the stuff like we did for head, because if
    we ran out of head, we're already fixed tail }
  FTail := pMsg^.Prev;
  Result := nil;
end

else begin
  pMsg^.Next^.Prev := pMsg^.Prev;
  pMsg^.Prev^.Next := pMsg^.Next;
  Result := pMsg^.Next;
end; { node in the middle }

Dispose(pMsg);
end;

{ TGConnection }

constructor TGConnection.Create(ASockParent : TObject;
  ASocket : TCustomWinSocket; const AHost : string);
begin
  inherited Create;

  FSocketParent := ASockParent;
  FSocket := ASocket;
  FHost := AHost;
  FRecvBuf := TMemoryStream.Create;

  if Assigned(FSocket) then
    FSocket.Data := Self;
end;

destructor TGConnection.Destroy;
begin
  FSocketParent.Free;
  FRecvBuf.Free;

  inherited;
end;

function TGConnection.GetRecvBufHead: Pgnutella_header;
begin
  Result := Pgnutella_header(FRecvBuf.Memory);
end;

function TGConnection.GetRecvBufPos: PChar;
begin
  Result := PChar(Integer(RecvBuf.Memory) + FRecvBuf.Position);
end;

function TGConnection.ReadFromSocket: integer;
begin
  if Assigned(FSocket) then
    Result := FSocket.ReceiveBuf(FRecvBuf.Memory^,
      FRecvBuf.Size - FRecvBuf.Position)
  else
    Result := 0;
end;

(***
  Take the message in SrcGConn's receive buffer and put it
  put on our wire
***)
procedure TGConnection.TransmitMsg(SrcGConn: TGConnection);

```



```
var
  iBytesSent: integer;
begin
  { todo: check for errors }
  if Assigned(FSocket) then
    iBytesSent := FSocket.SendBuf(SrcGConn.RecvBuf.Memory^, SrcGConn.RecvBuf.Size);
end;

end.
```

EXHIBIT A



[about](#) [download](#) [support](#) [resources](#) [press](#) [contact](#)

News

01/26/01 -
LimeWire 1.1 released

Statistics

Current Hosts: 151165
Available Hosts: 21563

[Download LimeWire](#)

About LimeWire

[Gnutella is an open protocol](#)
[Any content provider is welcome](#)
[Gnutella is decentralized](#)
[Share any kind of file](#)

Community

[Network Health article](#)
[Today's Host Graph](#)
[Rolling Host Graph](#)
[Good Citizen Tips](#)

Help

[Understanding P2P file-sharing](#)
[Gnutella Links](#)
[LimeWire Screensavers](#)
[LimeWire Banners](#)

Resources

[Download the Client](#)
[Need a good host?](#)
[LimeWire Support documents](#)
[Frequently asked questions](#)
[Screenshots of LimeWire](#)
[New Features](#)

Links

[Gnutella Protocol](#)
[Developer Principles](#)

[top](#)

[home](#) | [about](#) | [download](#) | [support](#) | [resources](#) | [press](#) | [contact](#) | [legal](#) | [sitemap](#)

[https://web.archive.org/web/20000928231323/http://docs.real.com/docs/plyrpl
us50.pdf](https://web.archive.org/web/20000928231323/http://docs.real.com/docs/plyrpl
us50.pdf)

© 1997 RealNetworks, Inc. All rights reserved.

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of RealNetworks, Inc. Your rights to the software and this documentation are governed by the accompanying software license agreement.

RealAudio is a registered trademark of RealNetworks, Inc. RealNetworks, RealVideo, RealPlayer, RealMedia and the Real logo are trademarks of RealNetworks, Inc.

Internet Explorer, Windows, Windows 95 and Windows NT are registered trademarks of Microsoft Corporation.

Apple and Macintosh are registered trademarks of Apple Computer, Inc.

Netscape, Netscape Navigator and Netscape Communicator are trademarks of Netscape Communications Corporation.

Any other trademarked products or companies referred herein are included for purposes of identification only, and without any intent to infringe.

RealNetworks, Inc.
1111 Third Avenue, Suite 2900
Seattle, WA 98101

<http://www.real.com>

Contents

Introduction to RealPlayer Plus 5.0	1
About this Manual	2
Documentation Conventions	2
Other Resources	2
RealPlayer Plus Components	3
Installing RealPlayer Plus	4
Windows 95 and Windows NT	4
Windows 3.1 and Windows 95 (using 16-bit Winsock)	6
Macintosh	8
PlusZone	10
Overview of RealPlayer Plus	11
Status Bar Icons	15
RealPlayer Plus Menus	16
File Menu	16
View Menu	18
Clip Menu	21
Sites Menu	22
Help Menu	24
Using RealPlayer Plus	26
Starting a Clip	26
Pausing a Clip	27
Stopping a Clip	27
Playing Clips in PerfectPlay Mode	27
Muting a Clip	28
Recording a Clip	28
Changing the Size of the RealPlayer Plus Window	30
Changing the Size of the Image Area	32
Setting a Preset Button	34
Selecting a Preset Button	34
Scanning RealMedia Content	35
Setting Your Destination Buttons	36
Setting Preferences	37
General Preferences	38
Connection Preferences	40
Transport Preferences	42

REALPLAYER PLUS GETTING STARTED GUIDE

Specify Transports Window	44
Proxy Preferences	46
Advanced Preferences	48
Language Preferences	51
Upgrade Preferences	53
Presets Preferences.....	55
Troubleshooting	57
I installed Internet Explorer after installing RealPlayer Plus and lost my RealPlayer Plus. What do I do?	57
How do I get Full Screen to work?	58
Why doesn't my RealPlayer Plus have a Volume Control slider?.....	58
How do I configure my browser to support RealMedia files?	58
Why can't I record from some sites?.....	59
Why can't I play clips on my computer at work?	59
What's wrong when the playback is poor?	59
Index	61

Introduction to RealPlayer Plus 5.0

Welcome to RealPlayer™ Plus 5.0 by RealNetworks. RealPlayer Plus gives you quality “one-button” access to the news and entertainment that you want.

Using RealPlayer Plus, you can listen to and view thousands of hours of live and pre-recorded clips, including sporting events, live radio stations, news, music, and lectures. RealPlayer Plus allows you to quickly and easily preset your favorite channels, scan for new stations, and enhance your control by getting better audio and video quality.

If you have not already installed RealPlayer Plus, please do so now. The installation takes just a couple of minutes. Installation instructions start on page 4.

For more information, refer to our Web site: <http://www.real.com>.

About this Manual

This documentation provides instructions for the installation, operation, and customization of RealPlayer Plus. It assumes that you have a basic knowledge of your operating system and your World Wide Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

Documentation Conventions

The following conventions are used in this manual:

Bold text indicates the names of commands you should select and buttons you should click.

Italic text indicates a URL (Internet address) you can access through your browser.

Monotype text indicates messages that appear on your computer screen.

SMALL CAPS text indicates a button on your keyboard.

Other Resources

For the latest information on our products, visit the RealNetworks' home page at:

<http://www.real.com>

Player Plus specific information, a technical knowledge base, a frequently asked questions list, and a technical library are available from the PlusZone at:

<http://pluszone.real.com/>

Please refer to the enclosed Technical Support Card for more information on contacting RealNetworks Technical Support.

RealPlayer Plus Components

In addition to RealPlayer Plus, several different special-purpose RealPlayer Plus components are installed on your computer. The components work with different elements within World Wide Web pages to deliver sound and video in a variety of ways. The RealPlayer Plus components are:

RealPlayer Plug-in	The Plug-in provides access to RealMedia programming and extends the audio and video capabilities of your Netscape Navigator browser 2.0 or later or Microsoft Internet Explorer browser 2.0 or later.
RealPlayer Plus Control for ActiveX	The RealPlayer Control provides access to RealMedia programming and extends the audio and video capabilities of your Microsoft Internet Explorer browser and other applications that support embedded ActiveX controls.
RealAudio Xtra for Shockwave	The RealAudio Xtra for Shockwave provides RealAudio programming within a Macromedia Shockwave movie.

Installing RealPlayer Plus

It is strongly recommended that you use at least a 28.8 Kbps modem with RealPlayer Plus. A 14.4 Kbps modem limits your ability to access much of the higher-quality RealMedia content.

You can improve the playback quality of some content by enabling the PerfectPlay mode. Refer to "Playing Clips in PerfectPlay Mode" on page 27 for more information.

Windows 95 and Windows NT

System and hardware requirements:

- Windows 95 or Windows NT 3.51 or later
- CPU:
 - Audio: 486 DX / 66 MHz minimum
 - Video: Pentium / 75 MHz minimum
- RAM Requirements:
 - Audio: at least 8 MB recommended
 - Video: at least 16 MB recommended
- Minimum of 2 MB of free hard-disk space
- CD-ROM drive
- 14.4 Kbps modem (28.8 Kbps or faster connection for playing video)
- Web browser and Internet connection

When you insert the RealPlayer Plus CD-ROM into your CD-ROM drive, the installation starts automatically.

RealPlayer Plus is installed in the C:\REAL\PLAYPLUS directory, unless you specify otherwise. The installation also installs several system files into the C:\WINDOWS\SYSTEM directory of your hard drive.

Note: For a complete list of the files that are installed, refer to the readme.htm file found in the C:\REAL\PLAYPLUS directory after the player is installed.

If the installation does not start automatically, perform the following steps:

To manually install RealPlayer Plus:

1. Close any open applications or programs in Windows.
2. Insert the CD-ROM into your CD-ROM drive.
3. Click **Run** on the Start menu.
4. Enter **CD Drive:\setup.exe** and click the **OK** button. (For example: **D:\setup.exe**) The RealPlayer Plus installation begins. Follow the instructions in each of the screens that follow.
5. When the installation is complete, you receive a message about visiting the PlusZone, the exclusive web site for RealPlayer Plus users. For more information refer to "PlusZone" on page 10.

When the installation is complete, a shortcut for RealPlayer Plus is added to your Windows 95 or Windows NT desktop, and a RealPlayer Plus icon is added to your system tray.

Windows 3.1 and Windows 95 (using 16-bit Winsock)

System and hardware requirements:

- Windows 3.1 or later or Windows 95 (using 16-bit Winsock)
- A Windows-compatible sound card is recommended (but not required)
- CPU:
 - Audio: 486 DX / 66 MHz minimum
 - Video: Pentium / 75 MHz minimum
- RAM Requirements:
 - Audio: at least 8 MB recommended
 - Video: at least 16 MB recommended
- Minimum of 2 MB of free hard-disk space
- CD-ROM drive
- 14.4 Kbps modem (28.8 Kbps or faster connection for playing video)
- Web browser and Internet connection
- Windows display set to at least 256 colors
- Video for Windows 1.1.e installed for video playback (Windows 3.1 only). The RealPlayer Plus searches for Video for Windows during installation. If it is not found, you may download the Video For Windows package from Microsoft's web site (<http://www.microsoft.com>).

RealPlayer Plus is installed in the C:\REAL\PLAYPLUS directory, unless you specify otherwise. The installation also installs several system files into the C:\WINDOWS\SYSTEM directory of your hard drive.

Note: For a complete list of the files that are installed, refer to the readme.htm file found in the C:\REAL\PLAYPLUS directory after the player is installed.

To install RealPlayer Plus:

1. Close any open applications or programs in Windows.
2. Insert the CD-ROM into your CD-ROM drive.
3. Click **Run** from the Program Manager File menu in Windows 3.1 or from the Start menu in Windows 95.
4. Enter **CD Drive:\setup.exe** and click the **OK** button. (For example: **D:\setup.exe**) The RealPlayer Plus installation begins. Follow the instructions in each of the screens that follow.
5. When the installation is complete, you receive a message about visiting the PlusZone, the exclusive web site for RealPlayer Plus users. For more information refer to "PlusZone" on page 10.

When the installation is complete, a RealPlayer group is added to your Windows 3.1 Program Manager or Windows 95 Programs menu.

Macintosh

System and hardware requirements:

- Macintosh PowerPC
- RAM Requirements:
 - Audio: at least 8 MB recommended
 - Video: at least 16 MB recommended
- Minimum of 2 MB of free hard-disk space
- CD-ROM drive
- 14.4 Kbps modem (28.8 Kbps or faster connection for playing video)
- Web browser and Internet connection

RealPlayer Plus is installed in the RealPlayer Plus folder on the Desktop, unless you specify otherwise. The RealPlayer Plus installation also installs several files into the System Folder of your computer.

Note: For a complete list of the files that are installed, refer to the ReadMe file found in the RealPlayer Plus folder after the player is installed.

INSTALLING REALPLAYER PLUS

To install RealPlayer Plus:

1. Close any open applications or programs.
2. Insert the CD-ROM into your CD-ROM drive.
3. Double-click the **RealPlayer Plus** CD-ROM icon.
4. Double-click **RealPlayer Plus Installer**. The RealPlayer Plus installation begins. Follow the instructions in each of the screens that follow.
5. When the installation is complete, you receive a message about visiting the PlusZone, the exclusive web site for RealPlayer Plus users. For more information refer to "PlusZone" on page 10.

PlusZone


A one-stop, centralized Web site for product information, online offers, and premium content, the PlusZone lets Plus customers learn about the product and find out about cool offers.

When RealPlayer Plus is successfully installed, the following message appears:



Figure 1: Visit PlusZone message

To visit the PlusZone:

- Click the  button. Your default World Wide Web browser opens and welcomes you to the PlusZone. From here you can visit the PlusZone Web site where you view special online offers for RealPlayer Plus customers, get your RealPlayer Plus questions answered, and view the RealPlayer Plus tutorial.

If you choose not to visit the PlusZone at this time, this message reappears the next time you open RealPlayer Plus. This enables you to visit at a later time. If you want to disable this message, check the **Please don't show this message again** box. If you want to access the PlusZone at a later time, you can do so by clicking the **customize your PlusZone page** link on the ReadMe.htm file in the same directory as the Player.

Note: If you upgrade or change your World Wide Web browser, open the ReadMe.htm file with your new browser and click the **customize your PlusZone page** link.

Overview of RealPlayer Plus

RealPlayer Plus enables you to listen and view files over the Internet or over a local area network in real-time, without downloading them to your hard drive. When you click a RealMedia link on a World Wide Web page, your RealPlayer Plus automatically opens and plays the file you have selected. In addition, you can pause, rewind, fast-forward, stop, and start the clip at the click of a button.

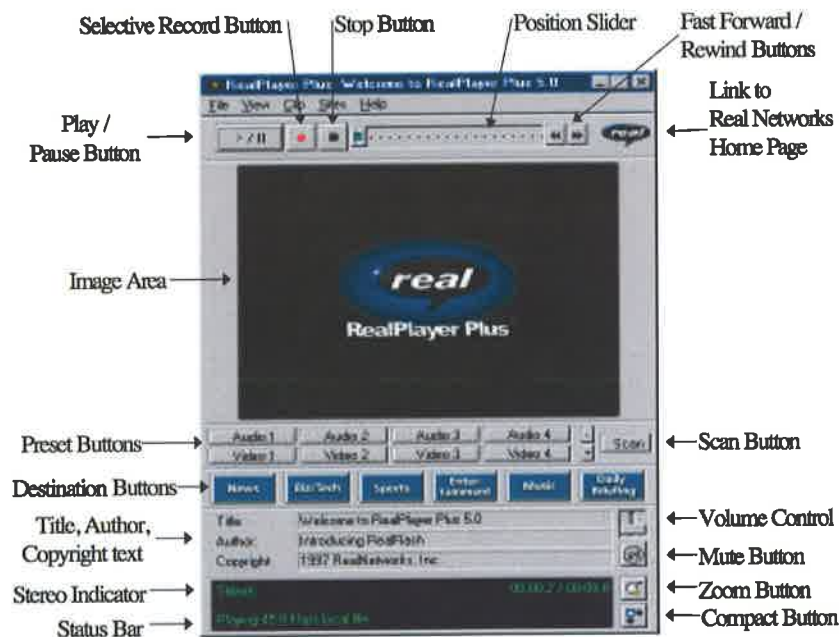






















Figure 2: Overview of RealPlayer Plus for Windows 95 and NT

Section	Description
Play/Pause Button	Click the  button to play or pause a clip, depending on the status of the clip.
Record Button	Click the  button to record the clip currently being played. Note: Content providers control whether or not you may record a clip.
Stop Button	Click the  button to stop the clip.
Position Slider	The slider moves as a clip is being played. You can fast forward or rewind to a different section of the clip by dragging the slider.
Fast Forward/Rewind Buttons	Click the  button to fast forward the clip or the  button to rewind the clip.
Link to the RealNetworks Home Page	If you are connected to the Internet, click the  button to link to the RealNetworks Home Page.
Image Window	If you are playing a video clip, the image appears in this window.
Preset Buttons	You can set these buttons to automatically link to a specific site and play a RealMedia clip. The URL location and the label of the Preset buttons are customized on the Preset tab of the Preferences window. You can also set a Preset button to the current clip by clicking with the right mouse button (Windows) or clicking and holding (Macintosh) the Preset button, then selecting Set Preset to (current clip) .

Section	Description
Scan Button	The  button allows you to scan available content, similar to scanning stations on your car radio. For additional selections, click with the right mouse button (Windows) or click and hold (Macintosh) the Scan button, a pop-up menu appears from which you can select to scan sports, news or music stations, or to scan your presets.
Destination Buttons	Click these buttons to go to the Destination web site where you can select premium content. The buttons are programmed to the content you select.
Title, Author and Copyright Text	The content provider specifies the text in these fields; they normally describe the clip being played.
Volume Control	<p>The volume control indicates the volume at which the clip is being played. Increase the volume by dragging the control up, and decrease the volume by dragging the control down.</p> <p>Note: If your RealPlayer Plus does not have a volume control slider, it is because your sound card does not allow the Player to adjust the volume.</p>
Mute Button	Click the  button to silence the audio portion of the clip. The clip continues to play. Click this button again to hear the audio portion of the clip.
Status Bar	<p>The status bar displays information about the player and the clip.</p> <p>For more information, refer to “Status Bar Icons” on page 15.</p>

Section	Description
00:00.0/00:00.0	<p>The elapsed time (the length of time that the clip has been playing) and the total time (the length of the entire clip).</p> <p>Note: When playing a live stream, Total Time value is replaced by the word live.</p>
Status	<p>This is a one-line informational display that gives the current status of the Player, explains menu options, and displays the assigned URL for each Preset button. For example: Playing 19 Kbps local file.</p>
Zoom Button	<p>Click the  button to open a menu where you can choose to double the video image area, enlarge the image area to full screen (Windows 95 and NT only), or return the image area to original size.</p> <p>Note: If you are running RealPlayer on Windows 3.1 or Macintosh, simply click the  button to double the image area. Click this button again to return the image area to the original size.</p>
Compact Controls Button	<p>Click the  button to display a subset of the buttons. Click the  button to display all of the controls.</p>

Status Bar Icons

Section	Description
	Indicates the clip currently being played is in stereo.
	Indicates the clip currently being played is in mono.
	Green lightning bolt indicates some loss of playback quality due to quality of connection.
	Red lightning bolt indicates poor playback quality due to quality of connection.
	Indicates that RealPlayer Plus is buffering the stream.
	Indicates that the network is busy and the player is buffering more data.
	Indicates that RealPlayer Plus is recording the current file.
	Indicates that the current file cannot be recorded.

RealPlayer Plus Menus

RealPlayer Plus has five menus: File, View, Clip, Sites, and Help. These menus allow you to take full advantage of RealPlayer Plus features.

File Menu

Commands from the File menu allow you to play RealMedia files without using your World Wide Web browser to find files on the Internet, to enable PerfectPlay mode, and to exit RealPlayer Plus.



Figure 3: File menu

Menu Option	Description
Open Location...	Select this option to play RealMedia files through the Internet without using your World Wide Web browser. The Open Location dialog appears, enter the URL (Internet address) of an <i>.ra</i> , <i>.rm</i> , or <i>.ram</i> file.
Open File...	Select this option to play a local RealMedia file. When the Open File dialog appears, select an <i>.ra</i> , <i>.rm</i> , or <i>.ram</i> file stored on your computer.

REALPLAYER PLUS MENUS

Menu Option	Description
Open Recent	Select this option to view a list of recently played clips. You can select a clip from this list to play the clip again.
PerfectPlay Mode	<p>Select this option to switch to PerfectPlay mode; a check mark appears next to the option. This mode enables you to improve the playback quality by increasing the amount of the clip that is buffered. The time it takes to buffer the clip will be slightly longer, but sound quality, especially with 14.4 and 28.8 Kbps modems, should be improved. To return to Regular mode, select this option again; the check mark is removed.</p> <p>You cannot use PerfectPlay mode with a live stream.</p>
Close Window (Macintosh only)	Select this option to close a secondary window, such as the Connections Statistics window.
Exit/Quit	Select this option to exit RealPlayer Plus.

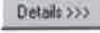
View Menu

Commands from the View menu allow you to control the appearance, to set user preferences, and view the performance of RealPlayer Plus.



Figure 4: View menu

Menu Option	Description
Normal	Select this option to display all buttons and controls; a mark appears next to the option.
Compact	Select this option to display only the image area (if playing a RealVideo or Flash clip), a subset of the buttons and a reduced Status bar.
Auto-Size	Select this option to display only the image area (if playing a RealVideo or Flash clip) when the cursor is not over the RealPlayer Plus window; a mark appears next to the option. When the cursor is on the window, the Compact controls are displayed.

Menu Option	Description
Presets & Scan	Select this option to display the Preset and Scan buttons; a check mark appears next to the option. Select the option again to hide the buttons; the check mark is removed.
Destinations	Select this option to display the Destinations; a check mark appears next to the option. Select the option again to hide the bar; the check mark is removed.
Status Bar	Select this option to display the Status Bar; a check mark appears next to the option. Select the option again to hide the bar; the check mark is removed.
Info & Volume	Select this option to display the volume control, and the title, author, and copyright information for the clip; a check mark appears next to the option. Select the option again to hide the information; the check mark is removed.
Preferences...	<p>Select this option to display the Preferences window for configuring RealPlayer Plus. For more information, refer to "Setting Preferences" on page 37.</p> <p>Note: This option is on the Edit menu in the Macintosh version of RealPlayer Plus.</p>
Statistics...	<p>Select this option to display the transmission quality information in the Connection Statistics dialog. The Connection Statistics dialog reports the performance of the Internet connection between RealServer and RealPlayer Plus. Refer to Figure 5 for an example of the Connection Statistics window.</p> <p>You can click the  button to see additional connection information.</p>

Menu Option	Description
Always on Top (Windows only)	Select this option to make RealPlayer Plus remain on top of all other windows, even when another window is active; a check mark appears next to the option. Always on Top is convenient for keeping RealPlayer Plus visible while you interact with your World Wide Web browser. Select this option again to allow other windows to open on top of RealPlayer Plus; the check mark is removed.

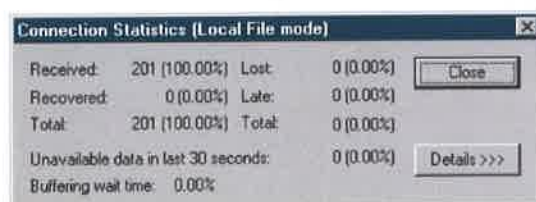


Figure 5: Connection Statistics window

Clip Menu

Some RealMedia files can include several clips played one after the other. Commands from the Clip menu allow you to move between clips in a multi-clip RealMedia (.ram) file and when scanning. This scanning is similar to using seek buttons on a CD player. If there is only one clip at the link you are accessing, these options are not available.

Previous Clip and Next Clip are also available from the shortcut menu that appears when you right-click (Windows) or click and hold (Macintosh) the mouse button in RealPlayer Plus window.

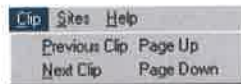


Figure 6: Clip menu

Menu Option	Description
Previous Clip	Select this option to play the previous clip in a multi-clip RealMedia (.ram) file or when scanning.
Next Clip	Select this option to play the next clip in a multi-clip RealMedia (.ram) file or when scanning.

Sites Menu

Commands from the Sites menu allow you to access various pages at the RealNetworks and Timecast World Wide Web sites. This feature works with popular World Wide Web browsers. If you are connected to the Internet, and your World Wide Web browser is not open, RealPlayer Plus opens it for you.

Note: You must be connected to the Internet to reach these sites.



Figure 7: Sites menu

Menu Option	Description
PlusZone	Select this option to access the PlusZone – the one-stop centralized web site for product information and premium content.
Real Home Page	Select this option to access the RealNetworks Home Page.
Real Content	Select these options to access sites that feature RealMedia.
Timecast: Your RealMedia Guide	Timecast contains a list of World Wide Web sites providing RealMedia content.
MusicNet	MusicNet contains a directory of RealMedia happenings in the music world.

REALPLAYER PLUS MENUS

Menu Option	Description
LiveConcerts.com	LiveConcerts.com is the premier site for live music events broadcast on the Internet using RealMedia.
Film.com	Film.com provides an unprecedented collection of material on film for a world-wide audience, offering breadth and depth, and bringing together critics, writers and movie buffs everywhere.
DailyBriefing.com	DailyBriefing.com is where you build your own customized video and audio daily newscast, choosing from the best news, sports, business, technology, entertainment, and music programs on the Web.
Customize RealPlayer Plus	Select these options to customize your RealPlayer Plus content buttons.
Destinations Tutorial	Select this option to access a page where you can learn how to set your preset and destinations buttons.
Customize Daily Briefing	Select this option to access a page where you can customize your Daily Briefing button.
Customize Destinations	Select this option to access a page where you can customize your destination buttons.
Customize Presets	Select this option to access a page where you can customize your preset buttons.

Help Menu

Commands from the Help menu provide you with online help and information about RealPlayer Plus.

The latest support and technical information is available at:

<http://service.real.com>

Please refer to the enclosed Technical Support Card for more information on contacting RealNetworks Technical Support.

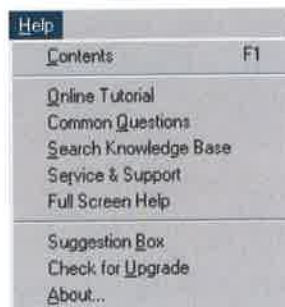


Figure 8: Help menu

Menu Option	Description
Contents	Select this option to open the online help for RealPlayer Plus. Note: The Macintosh version of RealPlayer Plus does not have online help.
Online Tutorial	Select this option to access the RealPlayer Plus tutorial, which describes the major features of the Player.

REALPLAYER PLUS MENUS


Menu Option	Description
Common Questions	Select this option to access the RealNetworks Frequently Asked Questions. You must be connected to the Internet for this option to work.
Search Knowledge Base	Select this option to access the RealNetworks Knowledge Base. You must be connected to the Internet for this option to work.
Service & Support	Select this option to access the RealNetworks main help page. You must be connected to the Internet for this option to work.
Full Screen Help (Windows 95 and NT only)	Select this option to access RealNetworks' Full Screen and DirectDraw information page.
Suggestion Box	Select this option to access the comments web page where you can enter suggestions or comments about the RealPlayer Plus.
Check for Upgrade	Select this option to query the RealNetworks web site for upgrade information. A dialog box tells you if there is an upgrade available.
About...	<p>Select this option to display RealPlayer Plus version and copyright information.</p> <p>Note: This option is on the Apple menu in the Macintosh version of RealPlayer Plus.</p>

Using RealPlayer Plus

Your RealPlayer Plus has several powerful features that allow you to make the most of the Internet.



Starting a Clip

There are several ways to start a RealMedia clip. Do one of the following:

- Click a RealMedia link on a World Wide Web page. Your World Wide Web browser opens RealPlayer Plus and the clip is streamed to RealPlayer Plus immediately. After storing a few seconds of the clip (buffering), the clip begins.
- Access a World Wide Web site or select an action (click a link, open an html page) that automatically plays a clip. Your World Wide Web browser opens RealPlayer Plus and the clip is streamed to RealPlayer Plus immediately. After storing a few seconds of the clip (buffering), the clip begins.
- Click one of the Preset buttons, or the  button on RealPlayer Plus. The clip is streamed to RealPlayer Plus immediately. After storing a few seconds of the clip (buffering), the clip begins.
- Click **Open File** or **Open Location** on the File menu on RealPlayer Plus. For more information, refer to "File Menu" on page 16.

Pausing a Clip

You can pause a clip while it is playing. If you pause a clip but do not resume playing after several minutes, the Server may terminate your connection. The length of time you can pause a clip without your connection terminating depends on the particular Server.

- Click the  button. This pauses the clip. If you want to resume the clip at the same point it was paused, click the  button.

Stopping a Clip

You can stop a clip while it is playing.

- Click the  button. The clip stops playing. To play the clip again from the beginning, click the  button.

Playing Clips in PerfectPlay Mode

PerfectPlay mode enables you to improve the playback quality by increasing the amount of the clip that is streamed to RealPlayer Plus before the clip begins playing. The time it takes to buffer the clip will be slightly longer, but playback quality, especially with 14.4 and 28.8 Kbps modems, should be improved.

You cannot use PerfectPlay mode with a live stream.

- Select **PerfectPlay Mode** from the File menu. Return to Regular mode by selecting this option again.

Note: Content providers may control whether PerfectPlay mode is available for a specific clip.


Muting a Clip

You can mute a clip while it is playing. The clip continues to play but does not play the audio.



- Click the  button. To hear the audio, click the button again.

Recording a Clip


You can record your favorite RealMedia clips for later playback from your computer. Keep in mind that not all content can be recorded. Content Providers control whether or not you may record a clip.

Note: If you attempt to record a clip that has this feature disabled, the  icon appears in the Status bar of RealPlayer Plus.



To start recording a clip during playback:

1. Click the  button. A Save As dialog box appears.
2. Select the filename and location on your computer where you want to save the file.
3. Click the **OK** button when you have finished entering the information. RealPlayer Plus begins recording. While RealPlayer Plus is recording, the  icon appears in the Status Bar of RealPlayer Plus.


Some RealMedia clips are several files played one after the other. If you chose to save a clip that is a compilation of several clips, a new Save As dialog box appears at the beginning of each new clip.

Note: Content providers control whether or not you may record a clip. If you attempt to record a clip that has this feature disabled, the  icon appears in the Status bar of RealPlayer Plus.

To set up recording a clip prior to playback:

1. Click the  button.
2. Open the clip. A Save As dialog box appears.
3. Select the filename and location on your computer where you want to save the file.
4. Click the **OK** button when you have finished entering the information. RealPlayer Plus begins recording and playing the selected clip. While the clip is recording, the  icon appears in the Status Bar of RealPlayer Plus.

Some RealMedia clips are several files played one after the other. If you chose to save a clip that is a compilation of several clips, a new Save As dialog box appears at the beginning of each new clip.

Note: Content providers control whether or not you may record a clip. If you attempt to record a clip that has this feature disabled, the  icon appears in the Status Bar of RealPlayer Plus.

To stop recording:


- Click the  button. RealPlayer Plus stops recording, but the clip continues to play.

To play a clip you recorded:

1. Select **Open File** from the File menu.
2. Select the *.ra*, *.rm*, or *.ram* file you recorded. RealPlayer Plus plays the file.

Changing the Size of the RealPlayer Plus Window

Do one of the following:

- Click the  button to display the RealPlayer Plus in Compact view. This view displays a subset of buttons, a smaller status bar and the image window (if not playing an audio-only clip).
- From the View menu, select the appropriate option to modify the size of the RealPlayer Plus window.
 - **Normal** displays all the controls and buttons.
 - **Compact** displays a select group of buttons:



- **Auto-Size** looks like Compact when playing an audio-only clip or when the cursor is over the window. For non audio-only clips, only an image window is displayed when the cursor is not over the window.



If the video or RealFlash clip won't fit on your RealPlayer screen using the current settings, RealPlayer will ask if you would like your settings changed automatically.

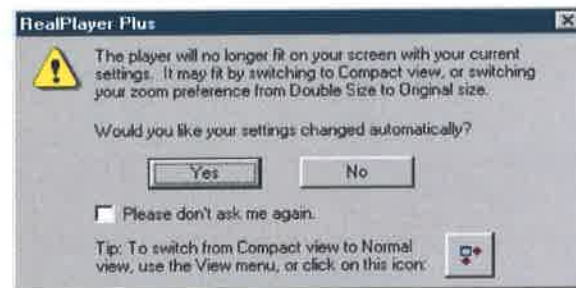


Figure 9: Change Current Settings Window

- Select **Yes** if you want RealPlayer to automatically change your settings.
- Select **No** if you don't want RealPlayer to automatically change your settings.
- If you want RealPlayer to always resize in this situation, select **Please don't ask me again**, and then select **Yes**.
- If you don't want RealPlayer to offer this option, select **Please don't ask me again**, and then select **No**.

Changing the Size of the Image Area

You can enlarge the size of a video or RealFlash clip image to double size. In addition, you can enlarge the size of a video clip image to full screen size (Windows 95 and NT only). However, these features increase the processing time required for playback. If your playback quality decreases or if your machine seems overloaded, you should return the clip image back to original size.

Note: In order to use the Full Screen feature, you must have Microsoft DirectX Tools with DirectDraw, and recent video drivers, installed on your machine. For information on downloading and installing the most recent version from Microsoft's web site, refer to "Troubleshooting" on page 55, or select **Full Screen Help** from the RealPlayer Help menu.


Before you can use the Full Screen feature, the Player will ask you to run a short test to see if your DirectX and video drivers are functioning properly. The test involves switching into Full Screen mode for 5 seconds. Then, you are asked to confirm whether you saw the RealPlayer logo in Full Screen mode. If yes, then the Player resumes the clip, switches to Full Screen mode, and you will not be asked to run the test in the future. If no, then you are asked if you would like to refer to RealNetworks' Full Screen and DirectDraw information page.

To increase the image area to double size:


1. Click the  button.
2. Select the **Double Size** option from the menu.

To return to original size from double size:

1. Click the  button.
2. Select the **Original Size** option.

Note: If you are running RealPlayer with Windows 3.1 or Macintosh, simply click the  button to double the image area. Click it again to return the video clip to original size.

To increase the image area for video clips to full screen (Windows 95 and NT only):

1. Click the  button.
2. Select the **Full Screen** option from the menu. The image will cover your screen entirely. Click anywhere on the screen to return to original size.

Setting a Preset Button

Preset buttons allow you to quickly jump to a RealMedia Internet address, much like the buttons on a radio jump to preset radio stations.

To set a Preset Button to the clip currently being played:

1. In the RealPlayer window, right-click the Preset button you want to set to the current clip and select **Set Preset to (current clip)**.
2. Type in a **Preset Label** for the Preset button and click **OK**.

To set a Preset button manually:

1. Choose the **Preferences** option from the **View** menu.
2. Choose the **Presets** tab.
3. Type in the full **URL** (Internet address) of the site for the Preset button.
4. You can also enter text for the **Label** of the Preset button.

To clear a Preset button:

- In the RealPlayer window, right-click the Preset button you want to clear and select **Clear Preset**.

To edit a Preset button:

- In the RealPlayer window, right-click the Preset button you want to edit and select **Edit Preset Button**. Type in any text changes to the **URL** or **Label**.

Selecting a Preset Button

To select a Preset button:

- Click the appropriate Preset button. RealPlayer Plus connects to the corresponding site and begins playing the clip.



To view other rows of Preset buttons:

- Click the up or down arrow next to the rows of presets to cycle through the preset rows.

Scanning RealMedia Content

The Scan feature allows you to sample RealMedia stations, similar to scanning stations on a radio. It plays each station for the amount of time specified on the Presets tab of the Preferences window and then continues to the next station.

To begin scanning stations:

- To scan a variety of available stations: click the  button. RealPlayer Plus scans and plays the next station. It plays the station for the amount of time specified on the Presets tab of the Preferences window and then continues to the next station.
- To select a category of stations: right-click (Windows) or click and hold (Macintosh) the  button. A pop-up menu appears, where you can select the category (for example: news, music, sports, international) or you can select to scan your Preset buttons. RealPlayer Plus scans for the next station that matches the category you selected.

To stop scanning when you find a station you like:

- Click the  button again.

To set a Preset button to the station currently being played:

- Right-click (Windows) or click and hold (Macintosh) on the Preset button you want to set to the current station and select **Set Preset**.

To return to a previous station:

- Select **Previous Clip** from the Clip menu or press PAGE UP (on Windows) / CMD-1 (on Macintosh).

To skip to the next station:

- Select **Next Clip** from the Clip menu or press PAGE DOWN (on Windows) / CMD-2 (on Macintosh).

To stop scanning and stop listening to the current station:

- Click the  button.

Setting Your Destination Buttons

Your RealPlayer Plus comes with six buttons that you can customize to access premium content with a push of a button.

To customize the Destination buttons:

- Click one of the buttons. RealPlayer Plus opens your default browser to the Customize Destination Buttons page where you can select specific content that you like. The buttons on your Player will then provide access to that content. Follow the directions on that web page.

To play Destination buttons content:

- Click the appropriate button. RealPlayer Plus plays the associated file.

Setting Preferences

You can customize RealPlayer Plus by changing the settings in the Preferences window (opened from the View menu). The Preferences control various aspects of RealPlayer applications functionality such as the number of clips to remember, network information, and proxy information. These preferences apply to all applications using RealPlayer technology.

Note: Macintosh preferences look slightly different, but the categories are the same.

General Preferences

The General tab in the Preferences window contains preferences for the number of clips saved on the Open Recent menu, Synchronized Multimedia support, and PerfectPlay mode.

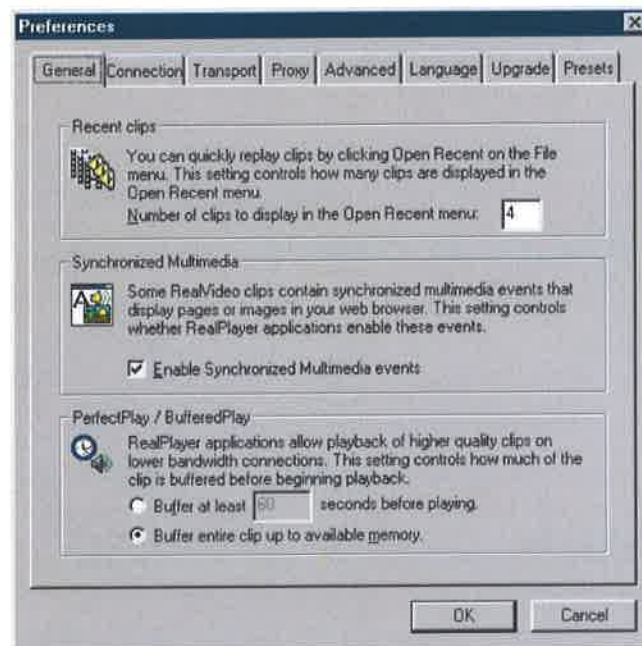


Figure 10: General tab in the Preferences window

SETTING PREFERENCES

Field	Description
Recent Clips	You can quickly replay clips by clicking Open Recent under the File menu. This setting controls how many clips are displayed in Open Recent.
Synchronized Multimedia	Some RealMedia clips contain synchronized multimedia events that display pages or images in your Web browser. This setting controls whether RealPlayer applications enable these events.
PerfectPlay/BufferedPlay	RealPlayer Plus allows playback of higher quality clips on lower bandwidth connections with the PerfectPlay feature. This setting controls how much of the clip is buffered before beginning playback.

Connection Preferences

The Connection tab in the Preferences window contains information about your bandwidth, statistics, and network timeout.

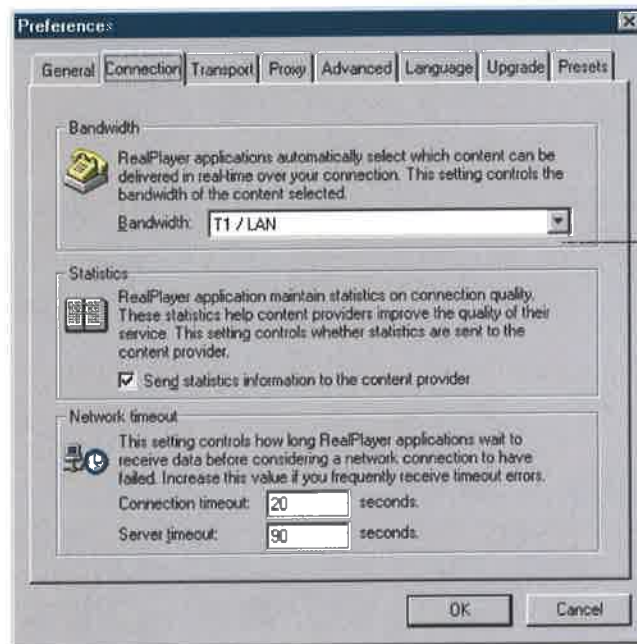


Figure 11: Connection tab in the Preferences window

SETTING PREFERENCES

Field	Description
Bandwidth	RealPlayer Plus automatically selects which content can be delivered in real-time over your connection. This setting controls the bandwidth of the content selected.
Statistics	<p>RealPlayer Plus maintains statistics on connection quality. These statistics help content providers improve the quality of their service. This setting controls whether statistics are sent to the content provider.</p> <p>Note: RealPlayer Plus does not send the content provider any information about your identity (such as your name or e-mail address).</p>
Network Timeout	<p>These settings control how long RealPlayer Plus waits to receive data before considering a network connection to have failed. Increase the values if you frequently receive timeout errors.</p> <p>Connection Timeout: This is the amount of time RealPlayer Plus will try a particular transport protocol.</p> <p>Server Timeout: This is the amount of time RealPlayer Plus will try to connect to a server.</p>

Transport Preferences

The Transport tab in the Preferences window allows you to customize your network connection for RealPlayer Plus. These preferences are for users behind a corporate firewall. Most likely, if you are connecting to the Internet through a commercial Internet Service Provider (such as AOL or MSN), you do not have to set any of these preferences.

The Transport preferences assume a basic knowledge of Internet protocols and firewalls. For more information about firewalls, refer to our World Wide Web site at <http://www.real.com/help/firewall/index.html>.

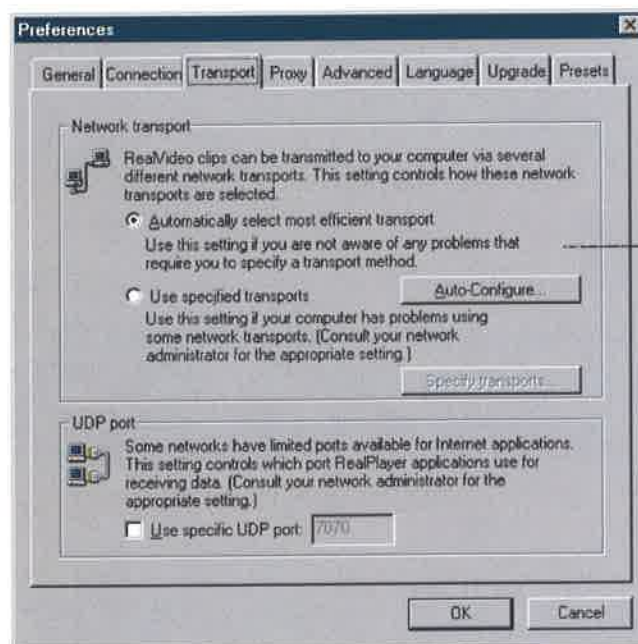


Figure 12: Transport tab in the Preferences window

SETTING PREFERENCES

Field	Description
Network Transport	RealMedia clips can be transmitted to your computer through several different network transport protocols. This setting controls how these network transport protocols are selected.
Automatically select most efficient transport	<p>Use this setting if you are not aware of any problems that require you to specify a transport method.</p> <p>Click the Auto Configure button to have RealPlayer Plus automatically configure itself to use the most efficient mode of transport for your connection.</p>
Use specified transports	<p>Use this setting if your computer has problems using some network transports. Consult your network administrator for the appropriate setting.</p> <p>Click the Specify Transports button to access the Specify Transport Settings window.</p>
UDP Port	<p>Some networks have limited ports available for Internet applications. This setting controls which port RealPlayer applications use for receiving data. Consult your network administrator for the appropriate setting.</p> <p>Note: This setting does not affect TCP packets that can be received only through Port 7070.</p>

Specify Transports Window

The Specify Transports window allows you to select the different modes of network transports that you can receive. Consult your network administrator for the appropriate setting.

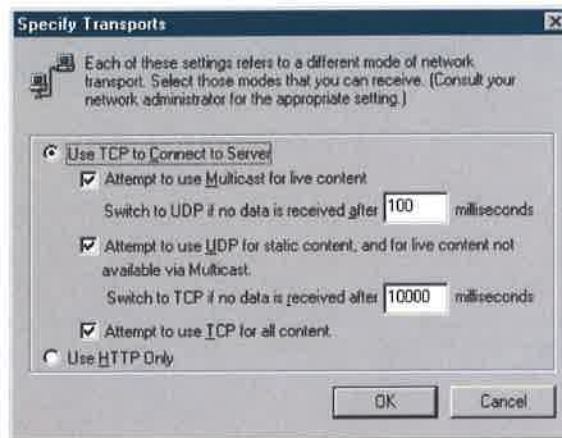


Figure 13: Specify Transports window

Field	Description
Use TCP to Connect to Server	Select this option if you want RealPlayer Plus to use TCP for the initial connection to RealServer.
Attempt to use Multicast for live content	Select this option if you want RealPlayer Plus to attempt to receive live content through multicast. If you select this option, indicate how many milliseconds the Player should attempt to use this type of transport before switching to UDP.

SETTING PREFERENCES

Field	Description
Attempt to use UDP for static content and for live content not available via Multicast	<p>Select this option if you want RealPlayer Plus to attempt to use UDP for static content and live content not available through multicast.</p> <p>If you select this option, indicate how many milliseconds the Player should attempt to use this type of transport before switching to TCP.</p>
Attempt to use TCP for all content	Select this option if you want RealPlayer Plus to use TCP for all content.
Use HTTP Only	<p>Select this option if you are behind a firewall and cannot receive data through TCP. All data will be streamed through HTTP.</p> <p>Note: You may not be able to receive some content if you select this option.</p>

Proxy Preferences

A proxy is a server that acts as an intermediary between a company's protected LAN and the Internet. The proxy server ensures that all data transmission between the Internet and a user on the LAN is authorized. RealPlayer Plus can be configured to work with proxy servers by setting the proxy preferences.

During the RealPlayer Plus installation, the Setup program looks for HTTP proxy information in your browser's preferences. If it finds proxy information, it displays this window with the information it found. You can verify that it is correct or modify it as necessary.

Note: For more information about proxies and firewalls, refer to our World Wide Web site at <http://www.real.com/help/firewall/index.html>.

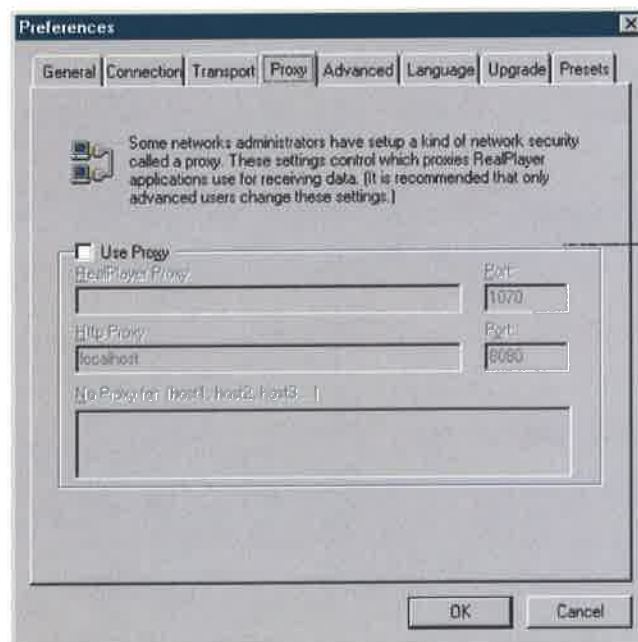


Figure 14: Proxy tab in the Preferences window

SETTING PREFERENCES

Field	Description
Use Proxy	<p>Click this box if RealPlayer Plus should connect to the Server using a proxy; a check mark appears in the field. Otherwise, do not click this box.</p> <p>Note: If you are behind a firewall and do not know the host name or port number of your proxy server, ask your system administrator.</p>
RealPlayer Proxy	<p>Enter the domain name or IP address of the RealPlayer Plus proxy server and port number that RealPlayer Plus should use to communicate with the proxy server.</p> <p>Note: For information on RealPlayer Proxy, read http://www.real.com/help/firewall/index.html</p>
HTTP Proxy	<p>Enter the domain name or IP address of the HTTP proxy server and port number that RealPlayer Plus should use to communicate with the proxy server. Check your World Wide Web browser's proxy configuration.</p>
No Proxy for: [Host1, Host2...]	<p>If you have specified that RealPlayer Plus should be connected to the Server using a proxy, you can specify certain servers that are inside your corporate firewall (domain names or IP addresses) with which you do not need to communicate using the proxy server. Enter these hosts in this field. This field is available only if you have enabled the Use Proxy option.</p> <p>Your system administrator can provide you with this information, if appropriate.</p>

Advanced Preferences

Options on the Advanced tab of the Preferences window override RealPlayer defaults for sound card compatibility, for CPU and memory requirements, for Zoom specifications, and for system tray use.

Note: RealPlayer Plus for Windows 3.1 does not have a Zoom specifications option or System Tray option. RealPlayer Plus for Macintosh does not have a System Tray option or Sound Card Compatibility option.

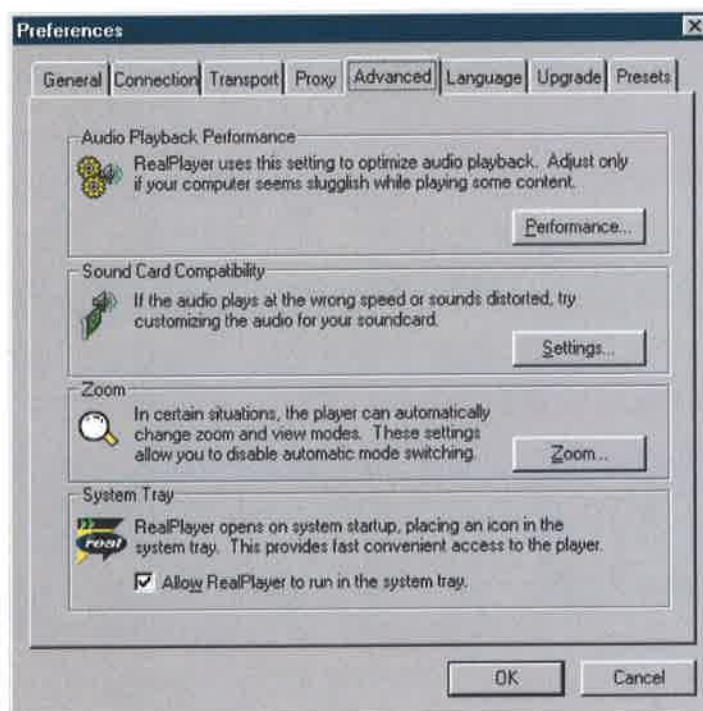


Figure 15: Advanced tab in the Preferences window

SETTING PREFERENCES

Field	Description
Audio Playback Performance	RealPlayer applications use this setting to optimize the performance of clip playback. Change this performance setting only if your computer seems sluggish or has problems while playing some audio content.
Sound Card Compatibility (Windows only)	Modify sound card settings to overcome audio problems.
Disable 16-bit sound (use 8-bit only)	Use this setting if you have an older sound card and clips sound scratchy or distorted.
Disable custom sampling rates	Use this setting if clips play at the wrong speed or sound scratchy or distorted. (Requires a floating point coprocessor.)
Zoom (not available in Windows 3.1)	Click the Zoom button to open a window where you can select Zoom specifications. Refer to Figure 16 for an example of the Zoom window.
System Tray (Windows 95 and NT only)	Select this option if you want RealPlayer Plus to stay resident on your system, creating an icon in your system tray. With this option selected, you will be able to quickly access RealPlayer Plus. By default, this option is selected.

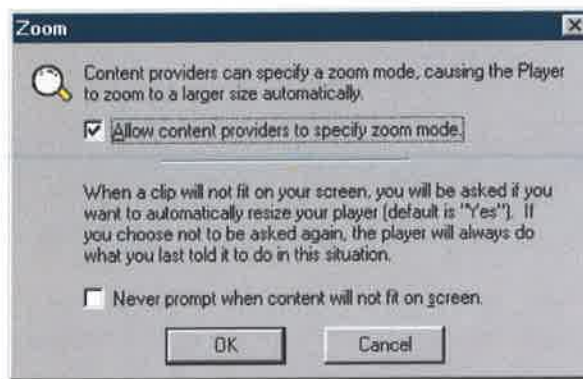


Figure 16: Zoom window

Field	Description
Allow content providers to specify zoom mode.	<p>Content providers can specify a Zoom mode, causing the Player to zoom to a larger size automatically. Select this option if you want content providers to specify zoom mode.</p> <p>By default, this option is selected.</p>
Never prompt when content will not fit on screen.	<p>When a clip will not fit on your screen, RealPlayer will ask if you want to automatically resize your window. Select this option if you don't want RealPlayer to prompt you when content will not fit on your screen using the current settings.</p> <p>For more information, refer to "Changing the Size of the RealPlayer Window" on page 30, and Figure 9, "Change Current Settings Window," on page 31.</p>

Language Preferences

The setting on the Language tab of the Preferences window determines the language used within RealPlayer Plus.

Note: RealPlayer Plus for the Macintosh does not have a Language Preferences option.

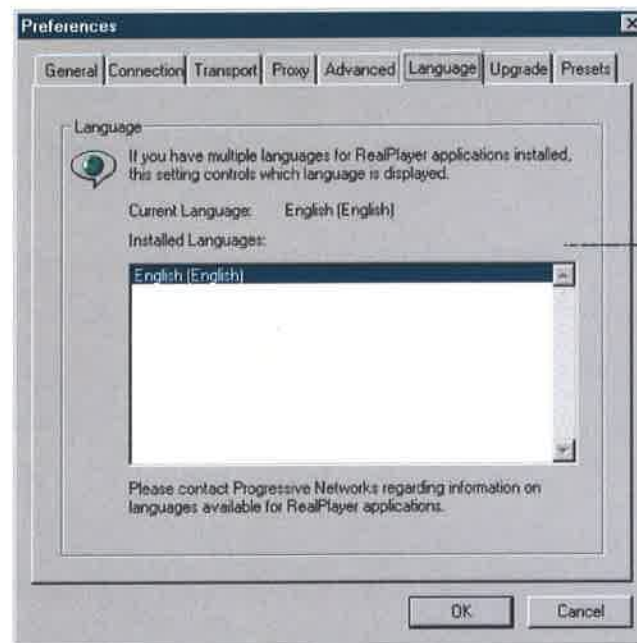


Figure 17: Language tab in the Preferences window

Field	Description
Current Language	This field displays the language being used within RealPlayer Plus.
Installed Languages	Select the language you want to use within RealPlayer Plus. Note: Please contact RealNetworks regarding information on languages available for RealPlayer Plus.

Upgrade Preferences

The setting on the Upgrade tab of the Preferences window determines if the RealPlayer Plus checks for the availability of newer version of the Player.

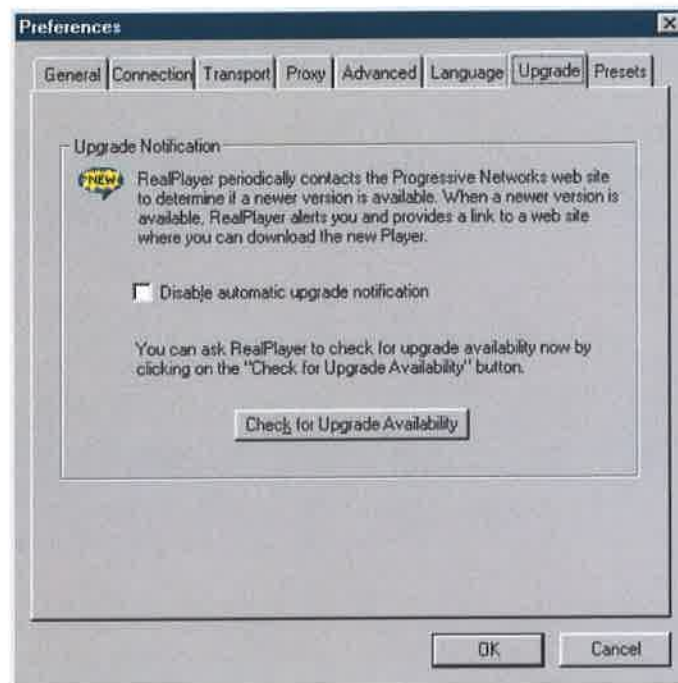


Figure 18: Upgrade tab in the Preferences window

Field	Description
Upgrade Notification	<p>RealPlayer Plus periodically contacts the RealNetworks Web site to determine if a newer version is available. When a newer version is available, RealPlayer Plus alerts you and provides a link to a Web site where you can download the new version.</p> <p>If you do not want RealPlayer Plus to check for upgrades, check the Disable Automatic Upgrade Notification button.</p>
Check for Upgrade Availability	Click this button to check for any available RealPlayer Plus upgrades.

Presets Preferences

Using the Presets tab of the Preferences window, you can modify the label of the Preset buttons and the URL corresponding to that Preset button. You can also set the length of time each clip is played when you scan.

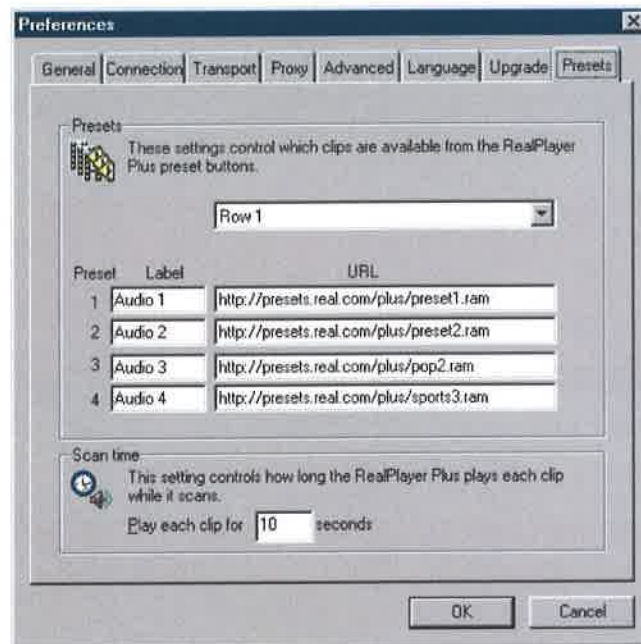


Figure 19: Presets tab in the Preferences window

Field	Description
Label	Enter the text you want to appear as the label of this Preset button.
URL	Enter the URL (Internet address) you want to play when this Preset button is clicked. You must enter the full URL. For example: <i>http://presets.real.com/plus/pop1 ram</i>
Scan Time	This setting controls how long RealPlayer Plus plays each clip as it scans.

Troubleshooting

The following are some common questions about RealPlayer Plus and recommended solutions.

Player Plus specific information, a technical knowledge base, frequently asked questions list, and technical library are available from the PlusZone at:

<http://pluszone.real.com/>

For the latest information on RealMedia products, visit the RealNetworks home page at:

<http://www.real.com>

Please refer to the enclosed Technical Support Card for more information on contacting RealNetworks Technical Support.

I installed Internet Explorer after installing RealPlayer Plus and lost my RealPlayer Plus. What do I do?

Microsoft Internet Explorer installs the RealAudio Player during its installation. This may overwrite your RealPlayer Plus. If this occurs, you need to re-install RealPlayer Plus after installing Internet Explorer.

How do I get Full Screen to work?

The Full Screen option is available for Windows 95 and Windows NT users only. If you are using one of these operating systems and still have problems with Full Screen (e.g. only a black screen appears), make sure that you have the following installed on your machine:

- **Microsoft DirectX Tools with DirectDraw**

You can download the most recent version and installation instructions from RealNetworks' DirectDraw support page (<http://service.real.com/fullscreen/default.html>). When you install DirectX, you must choose to upgrade your video drivers. You may lose old functionality of your drivers, however.

- **Updated video drivers**

You can find these either from the company that manufactured your video card or from RealNetworks' DirectDraw support page (<http://service.real.com/fullscreen/default.html>). If you update your video drivers, you may lose old functionality of your drivers.

Why doesn't my RealPlayer Plus have a Volume Control slider?


If the sound card installed in your computer does not support volume control, the Volume Control slider does not appear on RealPlayer Plus.

How do I configure my browser to support RealMedia files?

With most browsers, RealPlayer Plus installation configures the browser. If you change your browser and it stops recognizing RealMedia files, reinstall RealPlayer Plus. To manually configure your World Wide Web browser, refer to the following page:

<http://service.real.com/help/faq/index.html>

Why can't I record from some sites?

Content providers control whether or not you may record a clip. If you attempt to record a clip that has this feature disabled, the  icon appears in the Status Bar of RealPlayer Plus.

Why can't I play clips on my computer at work?

If you're on a local area network which is attached to the Internet, and are unable to play RealMedia files from remote Web sites, it's possible that your company's firewall is preventing the stream from reaching you. Users can receive live and on-demand clips from the Internet without exposing their company's network to security risks.

If your computer is behind a firewall, please see RealNetworks' firewall page for more information: <http://www.real.com/help/firewall/index.html>

What's wrong when the playback is poor?

The expected playback quality will vary depending on bandwidth and the encoding algorithm you are playing. If the quality is garbled, choppy, or skips, use the following to isolate and correct the problem.

If the sound quality was poor when the **Thankyou.rm** file played at the end of the RealPlayer Plus installation, you may have a sound card conflict. Try changing the settings on the Advanced tab in the Preferences window. Refer to "Advanced Preferences" on page 48.

If the playback quality was good when the **Thankyou.rm** file played at the end of the RealPlayer Plus installation, but is poor at a particular site, use the Connection Statistics window in RealPlayer Plus (select **Statistics** from the **View** menu) to check for lost packets. If the packet loss is high, it may be due to a busy network. Select **PerfectPlay Mode** from the **File** menu or connect to the site at a later time.

If the playback quality was good when the **Thankyou.rm** file played at the end of the RealPlayer Plus installation, but is poor at all sites, complete the following steps.

If the playback quality is poor on every site, do the following:

1. Check the actual modem connection speed. This may be shown in the lighted display for external modems, or through an information window if you have an internal modem (check the user manual for your modem). Sometimes service providers use a lower rate connection speed such as

14.4 Kbps, so even though you are dialing in on a 28.8 Kbps modem, you are only receiving data at 14.4 Kbps.

2. Contact RealNetworks' Technical Support by calling the support number on your Technical Support Card or sending e-mail using the form on [*http://service.real.com*](http://service.real.com).

Index

A

ActiveX, RealPlayer Plus
Control, 3
Advanced tab, 48

C

Clip Menu, 21
Clips
 muting, 28
 pausing, 27
 recording, 28
 starting, 26
 stopping, 27
Configuring browsers, 58
Connection Statistics window,
 20
Connection tab, 40

D

Destination Buttons, 36
Documentation Conventions, 2

F

File Menu, 16
Firewalls, 59

G

General tab, 38

H

Help Menu, 24

I

Image Area
 double size, 32
 full screen, 32
Installing, 4
 Macintosh, 8
 Windows 3.1/95 (16-bit
 Winsock), 6
 Windows 95/NT, 4
Internet Explorer, installation,
 57

L

Language tab, 51

M

Macintosh, installing, 8
Menus
 Clip, 21
 File, 16
 Help, 24
 Sites, 22
 View, 18
muting a clip, 28

P

Pausing a Clip, 27
PerfectPlay Mode, 17, 27
playback quality, 59
Plug-in, RealPlayer, 3
PlusZone, 10
Preferences
 general, 38
Preferences

- advanced, 48
- connection, 40
- languages, 51
- presets, 55
- proxy, 46
- setting, 37
- transport, 42
- upgrade, 53
- Preset button
 - changing, 34
 - selecting, 34
 - setting, 34
- Presets tab, 55
- Proxy tab, 46

R

- RealPlayer Plug-in, 3
- RealPlayer Plus
 - changing size of video image, 32
 - changing size of window, 30
 - overview, 11
 - using, 26
- RealPlayer Plus Control for ActiveX, 3
- Recording a Clip, 28

S

- Scanning RealMedia Clips, 35
- Setting Preferences, 37
- Shockwave Xtra, 3

- Sites Menu, 22
- Solving Problems, 57
- Specify Transports window, 44
- Starting a Clip, 26
- Stopping a Clip, 27
- Support, 57

T

- Technical Support, 2, 24, 57
- Transport tab, 42
- Troubleshooting, 57

U

- Upgrade tab, 53

V

- View Menu, 18
- volume control, missing, 58

W

- Windows
 - Connection Statistics, 20
 - Specify Transports, 44
- Windows 3.1/95 (16-bit Winsock), installing, 6
- Windows 95, installing, 4
- Windows NT, installing, 4

GnutellaDev

It's all about the Gnutelliums

POWERED BY WEGO.COM



Welcome guest!

[join group](#) | [log in](#)

View Protocol Pages

[INVITE OTHERS](#) | [HELP](#) | [FEEDBACK](#)

Web Pages

- Main
- Current problems
- Protocol
- Client Recommendations
- Charter
- All pages



Community

- Discussion board



Resources

- Shared files
- Web links
- Member directory

Search Web pages

 Submit
Query[Back to all pages](#)

PROTOCOL

Gnutella/0.4 protocol

The Gnutella protocol

Last update: 15 April 2000

Updated PUSH request routing instructions. Please comment. gene@wego.com

Notes

Everything is in network byte order unless otherwise noted. Byte order of the GUID is not important.

Apparently, there is some confusion as to what "\r" and "\n" are. Well, \r is carriage return, or 0x0d, and \n is newline, or 0x0a. This is standard ASCII, but there it is, from "man ascii".

Keep in mind that every message you send can be replied by multiple hosts. Hence, Ping is used to discover hosts, as the Pong (Ping reply) contains host information.

Throughout this document, the term server and client is interchangeable. Gnutella clients are Gnutella servers.

Thanks to capnbry for his efforts in decoding the protocol and posting it.

How GnutellaNet works

General description

GnutellaNet works by "viral propagation". I send a message to you, and you send it to all clients connected to you. That way, I only need to know about you to know about the entire rest of the network.

A simple glance at this message delivery mechanism will tell you that it generates inordinate amounts of traffic. Take for example the defaults for Gnutella 0.54. It defaults to maintaining 25 active connections with a TTL (TTL means Time To Live, or the number of times a message can be passed on before it "dies"). In the worst of worlds, this means 25^7 , or 6103515625 (6 billion) messages resulting from just one message!

Well, okay. In truth it isn't that bad. In reality, there are less than two thousand Gnutella clients on the GnutellaNet at any one time. That means that long before the TTL expires on our hypothetical message, every client on the GnutellaNet will have seen our message.

Obviously, once a client sees a message, it's unnecessary for it to process the message again. The original Gnutella designers, in recognition of this, engineered each message to contain a GUID (Globally Unique Identifier) which allows Gnutella clients to uniquely identify each message on the network.

So how do Gnutella clients take advantage of the GUID? Each Gnutella client

maintains a short memory of the GUIDs it has seen. For example, I will remember each message I have received. I forward each message I receive as appropriate, unless I have already seen the message. If I have seen the message, that means I have already forwarded it, so everyone I forwarded it to has already seen it, and so on. So I just forget about the duplicate and save everyone the trouble.

Topology

The GnutellaNet has no hierarchy. Every server is equal. Every server is also a client. So everyone contributes. Well, as in all egalitarian systems, some servers are more equal than others. Servers running on fast connections can support more traffic. They become a hub for others, and therefore get their requests answered much more quickly. Servers on slow connections are relegated to the backwaters of the GnutellaNet, and get search results much more slowly. And if they pretend to be fast, they get flooded to death.

But there's more to it than that.

Each Gnutella server only knows about the servers that it is directly connected to. All other servers are invisible, unless they announce themselves by answering to a PING or by replying to a QUERY. This provides amazing anonymity.

Unfortunately, the combination of having no hierarchy and the lack of a definitive source for a server list means that the network is not easily described. It is not a tree (since there is no hierarchy) and it is cyclic. Being cyclic means there is a lot of needless network traffic. Clients today do not do much to reduce the traffic, but for the GnutellaNet to scale, developers will need to start thinking about that.

Connecting to a server

After making the initial connection to the server, you must handshake. Currently, the handshake is very simple. The connecting client says:

```
GNUTELLA CONNECT/0.4\n\n
```

The accepting server responds:

```
GNUTELLA OK\n\n
```

After that, it's all data.

Downloading from a server

Downloading files from a server is extremely easy. It's HTTP. The downloading client requests the file in the normal way:

```
GET /get/1234/strawberry-rhubarb-pies.rcp HTTP/1.0\r\n
Connection: Keep-Alive\r\n
Range: bytes=0-\r\n
\r\n
```

As you can see, Gnutella supports the range parameter for resuming partial downloads. The 1234 is the file index (see HITS section, below), and "strawberry-rhubarb-pies.rcp" is the filename.

The server will respond with normal HTTP headers. For example:

```
HTTP 200 OK\r\n
Server: Gnutella\r\n
Content-type: application/binary\r\n
Content-length: 948\r\n
\r\n
```

The important bit is the "Content-Length" header. That tells you how much data to expect. After you get your fill, close the socket.

Header

bytes	summary	description												
0-15	Message identifier	This is a Windows GUID. I'm not really sure how globally-unique this has to be. It is used to determine if a particular message has already been seen.												
16	Payload descriptor (function identifier)	<table><tr><th>Value</th><th>Function</th></tr><tr><td>0x00</td><td>Ping</td></tr><tr><td>0x01</td><td>Pong (Ping reply)</td></tr><tr><td>0x40</td><td>Push request</td></tr><tr><td>0x80</td><td>Query</td></tr><tr><td>0x81</td><td>Query hits (Query reply)</td></tr></table>	Value	Function	0x00	Ping	0x01	Pong (Ping reply)	0x40	Push request	0x80	Query	0x81	Query hits (Query reply)
Value	Function													
0x00	Ping													
0x01	Pong (Ping reply)													
0x40	Push request													
0x80	Query													
0x81	Query hits (Query reply)													
17	TTL	Time to live. Each time a message is forwarded its TTL is decremented by one. If a message is received with TTL less than one (1), it should not be forwarded.												
18	Hops	Number of times this message has been forwarded.												
19-22	Payload length	The length of the ensuing payload.												

Payload: ping (function 0x00)

No payload

Routing instructions

Forward PING packets to all connected clients. Most other documents state that you should not forward packets to their originators. I think that's a good optimization, but not a real requirement. A server should be smart enough to know not to forward a packet that it originated.

A cursory analysis of GnutellaNet traffic shows that PING comprises roughly 50% of the network traffic. Clearly, this needs to be optimized. One of the problems with clients today is that they seem to PING the network periodically. That is indeed necessary, but the frequency of these "update" PINGs can be drastically reduced. Simply watching the PONG messages that your client routes is enough to capture lots of hosts.

One possible way to really reduce the number of PINGs is to alter the protocol to support PING messages which includes PONG data. That way you need only wait for hosts to announce themselves, rather than discovering them yourself.

Payload: pong (query reply) (function 0x01)

bytes	summary	description
0-1	Port	IPv4 port number.
2-5	IP address	IPv4 address. x86 byte order! Little endian!
6-9	Number of files	Number of files the host is sharing.
10-13	Number of kilobytes	Number of kilobytes the host is sharing.

Routing instructions

Like all replies, PONG packets are "routed". In other words, you need to forward this packet only back down the path its PING came from. If you didn't see its PING, then you have an interesting situation that should never arise. Why? If you didn't see the PING that corresponds with this PONG, then the server sending this PONG routed it incorrectly.

Payload: query (function 0x80)

bytes	summary	description
0-1	Minimum speed	The minimum speed, in kilobytes/sec, of hosts which should reply to this request.
2+	Search criteria	Search keywords or other criteria. NULL terminated.

Routing instructions

Forward QUERY messages to all connected servers.

Payload: query hits (query reply) (function 0x81)

bytes	summary	description
0	Number of hits (N)	The number of hits in this set. See "Result set" below.
1-2	Port	IPv4 port number.
3-6	IP address	IPv4 address. x86 byte order! Little endian!
7-10	Speed	Speed, in kilobits/sec, of the responding host.
11+	Result set	There are N of these (see "Number of hits" above).

bytes	summary	description
0-3	Index	Index number of file.
4-7	Size	Size of file in bytes.
8+	File name	Name of file. Terminated by double-NULL.

Last 16 bytes Client identifier GUID of the responding host. Used in PUSH.

Routing instructions

HITS are routed. Send these messages back on their inbound path.

Payload: push request (function 0x40)

bytes	summary	description
0-15	Client identifier	GUID of the host which should push.
16-19	Index	Index number of file (given in query hit).
20-23	IP address	IPv4 address to push to.
24-25	Port	IPv4 port number to push to.

Routing instructions

Forward PUSH messages only along the path on which the query hit was delivered. If you missed the query hit then drop the packet, since you are not instrumental in the delivery of the PUSH request.

Need some feedback on this one.

Copyright © 1999 Wego.com Incorporated. All rights reserved.

[Contact WeGo](#)



Gnutella and Freenet Represent True Technological Innovation

by [Andy Oram](#)
05/12/2000



The computer technologies that have incurred the most condemnation recently -- [Napster](#), [Gnutella](#), and [Freenet](#) -- are also the most interesting from a technological standpoint. I'm not saying this to be perverse. I have examined these systems' architecture and protocols, and I find them to be fascinating. Freenet emerged from a bona fide, academically solid research project, and all three sites are worth serious attention from anyone interested in the future of the Internet.

In writing this essay, I want to take the hype and hysteria out of current reports about Gnutella and Freenet so the Internet community can evaluate them on their merits. This is a largely technical article; I address the policy debates directly in a companion article, [The Value of Gnutella and Freenet](#). I will not cover Napster here because its operation has received more press. It's covered in "[Napster: Popular Program Raises Devilish Issues](#)" by Erik Nilsson, and frankly, it is less interesting and far-reaching technically than the other two systems.

In essence, Gnutella and Freenet represent a new step in distributed information systems. Each is a system for searching for information; each returns information without telling you where it came from. They are innovative in the areas of distributed information storage, information retrieval, and network architecture. But they differ significantly in both goals and implementation, so I'll examine them separately from this point on.

Gnutella basics

Each piece of Gnutella software is both a server and a client in one, because it supports bidirectional information transfer. The Gnutella developers call the software a "servent," but since that term looks odd I'll stick to "client." You can be a fully functional Gnutella site by installing any of several available clients; lots of different operating systems are supported. Next you have to find a few sites that are willing to communicate with you: some may be friends, while others may be advertised Gnutella sites. People with large

Related Articles:

[Napster and MP3: La
Revolucion or La
Larceny?](#)

[Music Industry Turns
Heat on Net Music
Pirates](#)

[Why the RIAA is
Fighting a Losing Battle](#)

[Napster: Popular
Program Raises Devilish
Issues](#)

[Why the RIAA Still
Stands a Chance](#)

computers and high bandwidth will encourage many others to connect to them.

You will communicate directly only with the handful of sites you've agreed to contact. Any material of interest to other sites will pass along from one site to another in store-and-forward fashion. Does this sound familiar, all you grizzled, old UUCP and Fidonet users out there? The architecture is essentially the same as those unruly, interconnected systems that succeeded in passing Net News and e-mail around the world for decades before the Internet became popular.

But there are some important differences. Because Gnutella runs over the Internet, you can connect directly with someone who's geographically far away just as easily as with your neighbor. This introduces robustness and makes the system virtually failsafe, as we'll see in a minute.

Evil or Just Controversial?:

Open Source software such as Gnutella and Freeware are spreading as quickly as a virus. But are they really so unhealthy? Andy Oram points out the advantages--and disadvantages--of controversial technologies in this week's edition of [Platform Independent](#) on Web Review.

Second, the protocol for obtaining information over Gnutella is a kind of call-and-response that's more complex than simply pushing news or e-mail. Figure 1 shows the operation of the protocol. Suppose site A asks site B for data matching "MP3." After passing back anything that might be of interest, site B passes the request on to its colleague at site C -- but unlike mail or news, site B keeps a record that site A has made the request. If site C has something matching the request, it gives the information to site B, which remembers that it is meant for site A and passes it through to that site.



Figure 1. How Gnutella retrieves information

I am tempted to rush on and describe the great significance of this simple system, but I'll pause to answer a few questions for those who are curious.

1. How are requests kept separate?

Each request has a unique number, generated from random numbers or semi-randomly from something unique to the originating site like an Ethernet MAC address. If a request goes through site C on to site

D and then to site B, site B can recognize from the identifier that it's been seen already and quietly drop the repeat request. On the other hand, different sites can request the same material and have their requests satisfied because each has a unique identifier. Each site lets requests time out, simply by placing them on a queue of a predetermined size and letting old requests drop off the bottom as new ones are added.

2. What form does the returned data take?

It could be an entire file of music or other requested material, but Gnutella is not limited to shipping around files. The return could just as well be a URL, or anything else that could be of value. Thus, people are likely to use Gnutella for sophisticated searches, ending up with a URL just as they would with a traditional search engine. (More on this exciting possibility later.)

3. What protocol is used?

Gnutella runs over HTTP (a sign of Gnutella's simplicity). A major advantage of using HTTP is that two sites can communicate even if one is behind a typical organization's firewall, assuming that this firewall allows traffic out to standard Web servers on port 80. There is a slight difficulty if a client behind a firewall is asked to serve up a file, but it can get by the firewall by issuing an output command called GIV to port 80 on its correspondent. The only show-stopper comes when a firewall screens out all Web traffic, or when both correspondents are behind typical firewalls.

4. How does the system stop searching?

Like IP packets, each Gnutella request has a time-to-live, which is normally decremented by each site until it reaches zero. A site can also drastically reduce a time-to-live that it decides is ridiculously high. As we will see in a moment, the time-to-live limits the reach of each site, but that can be a benefit as well as a limitation.

5. How is a search string like "MP3" interpreted?

That is the \$64,000 question, and leads us to Gnutella's greatest contribution.

Pages: 1, 2, 3, 4

[Next Page](#) ►

[CONTACT US](#) | [MEDIA KIT](#) | [PRIVACY POLICY](#) | [PRESS CENTER](#) | [JOBS](#) |

Copyright © 2000 O'Reilly and Associates, Inc. All Rights Reserved.
All trademarks and registered trademarks appearing on the O'Reilly Network are the property of their respective owners.

Archive of UserLand's first discussion group, started October 5, 1998.

Re: How does Gnutella play with WinAmp?

Author: Lawrence Lee
Posted: 7/14/2000; 11:48:19 PM
Topic: [How does Gnutella play with WinAmp?](#)
Msg #: 18644 (In response to [18643](#))
Prev/Next: [18643](#) / [18645](#)

I installed Gnutella and didn't notice any player connection (like Napster's library or right-click context menu), but the [Gnutella tutorial](#) mentioned the MP3 streaming in Gnutella:

"This only works for MP3 files and you must have WinAMP to begin with. All this does is put the MP3 file name in your WinAMP playlist so that you can list[en] to the song as it downloads."

From the #*gnutella* channel on [Efnet IRC](#) about the connection:

Lawrence: Can someone explain how Gnutella talks to Winamp if you stream a MP3?
Yytrium: it launches it with the name as a parameter
Yytrium: it doesn't do any talking

There are responses to this message:

- [Re: How does Gnutella play with WinAmp?](#), Dave Winer, 7/15/2000; 10:25:36 AM

This page was archived on 6/13/2001; 4:55:41 PM.

© Copyright 1998-2001 [UserLand Software](#), Inc.

Network Software Research Project Overview

A lot of the work done in the past two years in our department has helped create and advance the Imminet Product line. This portfolio of products form a complete content delivery solution, enabling smarter and faster networks. Please see the [Imminet Solutions page](#) for more information about the products. The products, which we continue to help in developing include:

- **WebCache** : The [Imminet WebCache](#) improves network response time, delivers fresh content and reduces server overload. These scalable, carrier-grade network caching appliances help sustain optimal network performance while enhancing the quality of service provided to customers. This performance improvement translates into bandwidth and server cost savings and fresher content for users. The WebCache was custom built with a proprietary, highly optimized file system, for extremely fast operation. Our research team continues to work with the Imminet development organization to advance and optimize the current product.
- **WebStream** : The Imminet WebStream is a streaming cache, and like the WebCache, translates into bandwidth and server cost savings by caching frequently viewed streaming media files at the edge of networks and close to end users. The current product can cache streams served in Real Networks Real Format. Windows Media format and Apple QuickTime format caching engines are being developed and will be available in 2001. Our research group has also created a streaming cache based on standards such as MPEG-1 and RTP. Three papers have been written outlining this work: 1) [Caching Techniques for Streaming Multimedia over the Internet](#); 2) [Design and Implementation of a Caching System for Streaming Media over the Internet](#), and; 3) [Streaming and Broadcasting over the Internet](#).
- **WebDNS** : The [Imminet WebDNS](#) is currently in Alpha test. This product was co-developed by the Imminet product team, members of the [Mathematical Sciences Research Center](#), and members of our team. Based on patent-pending algorithms, the WebDNS cooperates with existing Domain Name System elements to direct incoming requests to the service node with a WebCache that is closest, in network proximity, to the client generating the request. A WebCache at the service node acts as a switched reverse proxy and delivers the requested content to the client.
- **WebServices** : The wide deployment of network edge proxy caches and their strategic role as a gateway between Web clients and Web content servers suggests to utilize them for services that go beyond conventional Web caching. The Internet as we know it is characterized by simple end-to-end semantics. Both content providers and Web users, however, are increasingly looking to the network for additional, intelligent services performed by middleware layers like proxy caches. The [WebServices](#) project seeks to augment the Imminet product line with a complete solution for intelligently creating value added services at the edge of the network. These services include virus scanning, ad insertion, language translation. Our group is very active in this area within the [IETF](#). Markus Hofmann and Andre Beck have submitted several Internet Drafts in this area.
 - iCAP Working Group: [ICAP the Internet Content Adaptation Protocol](#), Internet Draft: draft-opes-icap-00.txt, Work in Progress, November 2000.
 - A. Beck, M. Hofmann: , [PSRL - A Rule Specification Language for Proxy Services](#) Internet Draft: draft-beck-opes-psrl-00.txt, Work in Progress, November 2000.
 - A. Beck, M. Hofmann, M. Condry: [Example Services for Network Edge Proxies](#), Internet Draft: draft-beck-opes-esfne-01.txt, Work in Progress, November 2000.

Past Research Projects

[Reliable Multicast Transport Protocol](#) : RMTP is a reliable multicast transport protocol for the Internet. It provides sequenced, lossless delivery of a data stream from one sender to a group of receivers.

Standards Organizations and Alliances

Our group is very involved in outside activities that advance the state-of-the-art in content delivery and streaming media. We are active in the following groups:

- [IETF](#) : The Internet Engineering Task Force (IETF) is a large open international community of network designers,

EXHIBIT A

operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.

- [Broadband Content Delivery Forum](#) : The Broadband Content Delivery Forum (BCDF) is a unique collaborative effort of leading internet Infrastructure, Content and Service providers that are working together to accelerate the deployment of broadband content over the internet - enabling customers to self-select and access high quality, multimedia internet content anytime, anywhere.
- [Internet Streaming Media Alliance](#) : The goal of this alliance is to accelerate the adoption of open standards for streaming rich media - video, audio, and associated data - over the Internet.

Improving Gnutella's Ping/Pong Scheme

Christopher Rohrs, Lime Peer Technologies LLC (LimeWire)
with ideas from Vincent Falco, Free Peers Inc. (BearShare)

Abstract: we propose a revised ping/pong scheme for the Gnutella network. Our scheme helps clients connect to the network more easily, uses a small amount of bandwidth, and is compatible with older clients. The key idea is to cache pongs and send only the best N pongs to a client.

Introduction

Gnutella's ping and pong messages serve two functions:

1. Return IP addresses of other servents so you can establish more connections.
2. Provide a measure of the horizon size, i.e., the number of hosts and files searchable.

Unfortunately the current ping/pong scheme has two problems. First, it consumes an enormous amount of bandwidth—up to 50% of the total Gnutella messaging bandwidth. Secondly, most returned addresses are not reachable or not accepting incoming connections. For this reason, it is difficult to connect to the Gnutella network.

This document proposes an improved way of achieving the first function. Our method does not introduce any new messages and requires no modifications to the existing ping and pong message formats; it simply changes when they are sent and how they are interpreted. Our scheme has three properties:

1. **Efficiency.** The amount of bandwidth used for pings/pongs is capped at roughly 100 bytes/sec/connection, even in the presence of an attacker or older clients.
2. **Quality.** If a host A receives a pong from another host B, B was accepting incoming connections within the last 15 seconds.
3. **Fairness.** If a host sends a ping and has been playing by the rules, it is guaranteed a pong in response—unless there are really no hosts accepting incoming connections.

Our scheme does not attempt to measure the horizon accurately. As long as a servent receives one pong from each host in its horizon, pongs will consume a lot of bandwidth. As an example, receiving 4000 pongs would require $(4000 \text{ pongs}) \cdot (37 \text{ bytes/pong}) / (1 \text{ sec}/5000 \text{ bytes}) = 30 \text{ seconds}$ on a typical 56kb/sec modem. However, we do suggest some new ways to approximate the horizon in the appendix.

The key idea of our scheme is to cache pongs to avoid broadcasting pings too rapidly. We also introduce a technique called *ping multiplexing* to avoid additional broadcasts while the cache is filling. Additional refinements include limiting the number of pongs sent in response to a ping, and throttling incoming pings to defend against attackers and old clients. This entire scheme can be implemented easily and efficiently, and we give complete pseudocode at the end of this document.

Pong Limiting

Pong limiting is an easy way to reduce the number of pongs on the network. First of all, a host should only send a pong with its own address if that host is not firewalled and can currently accept incoming connections. This is critical to our scheme. Remember that we are not trying to provide information about the horizon size; we are only interested in efficiently helping people connect to the network quickly.

Second, a host should only route a fixed number of pongs in response to a ping. If everyone is using the above rule, picking 10 or so pongs should be enough to help hosts connect quickly. The question is which 10 pongs to send back. Pongs with lower hops values are more useful to unconnected clients trying to establish a permanent connection, as these pongs are younger. On the other hand, pongs with higher hops values are more useful to

neighbors trying to increase their horizon size through additional connections. Therefore a good approach is to try to distribute hops values equally and to send pongs from a variety of connections. In this case, that might mean sending one pong with a hops value of 1 from connection 1, one pong with a hops value of 2 from connection 2, etc. Clients may also strive to send different pongs to each connection in response to each ping.

Pong Caching

Pong limiting does not solve the efficiency problem if too many pings are being sent. Consider a single servent A with a stable set of good connections. Now another host B joins A and sends a ping. A broadcasts the ping to all hosts within its horizon and routes the pongs back. Now say that B disconnects and is replaced by another host C. As usual C send a ping to A. Unless a lot of time has elapsed since B sent its ping, the ping from C will travel to the same set of hosts, and A will route the same set of pongs back. So twice as many messages have been sent than necessary. More generally, any two pings that a hosts sees within a small period of time are likely to have the same effect.

We can take advantage of this by caching the most recent pongs and avoiding the broadcast of pings. In the above example, A caches the pongs it saw in response to B's ping and simply sends these back to C—without forwarding C's ping. This is illustrated to the right. Of course A will have to set the GUID of each pong that it sends to C. If every host on the network is caching pongs, the amount of traffic will be greatly reduced.

To ensure that hosts get fresh pongs, it is important that the cache expires every few seconds. To refresh its cache, a host pings all its neighbors every 3 seconds and stores all pongs sent in response. The TTL on this ping is the standard outgoing value, say 5 or 7. Broadcasting a ping every few seconds may sound like a crazy idea, but some quick analysis shows that very little bandwidth is actually used. The trick is to observe that hosts receiving a ping usually respond by simply dipping into their pong caches. In other words, a ping is rarely a true broadcast message.

Let the cache expiration time, in seconds, be T . Let N be the maximum number of pongs sent in response to a ping. Then at most one ping and N pongs are sent per connection per T seconds. Since a ping is 23 bytes and a pong is 37 bytes, the amount of bandwidth used per connection is $(23+N*37)/T$ bytes/sec. Assuming $T=3$ and $N=10$, that works out to only 131 bytes/sec/connection. So a typical modem user with 2 connections and a total bandwidth of 5 kbytes/sec will only have to devote 5% of his bandwidth to pings and pongs—an order of magnitude less than the current scheme. Reducing N or increasing T can further decrease bandwidth requirements. Furthermore, no pong will be older than $M*T$ seconds, where M is the maximum allowable TTL. For $M=5$, that works out to only 15 seconds.

In an earlier version of our scheme, if a host with an empty cache received a ping with TTL i , it refreshed the cache by broadcasting the ping with a TTL of $i-1$ —not M —and storing the results. If the host subsequently received a ping with a higher TTL, it would broadcast that ping immediately. We called this process a “TTL upgrade”. If there is little demand for pongs, the earlier demand-driven scheme is more efficient than our proposed scheme because only a small portion of the network is pinging at any given time. However, if pongs are in demand, it is less efficient because each node may send up to M pings (and $N*M$ pongs) per connection per T seconds. For this reason, our proposed scheme does not use TTL upgrades.

Ping Multiplexing

The above discussion assumes that hosts reply to a ping instantaneously. Obviously this is not so. The worst case occurs when the caches of all hosts expire at the same time. In this case, every ping results in a “cache miss”, so hosts must wait for pongs to be routed from their originator. If the latency per connection is not significantly smaller than the cache expiration time, a host may receive a pong shortly before its cache expires. For this reason, it is advisable that T be no smaller than 3 seconds. Furthermore, clients implementing message

prioritization should make pings and pongs a *higher* priority message than queries and replies.

Another difficulty arises when two pings come in quick succession. Consider three hosts A, B, and C, connected in a line as shown in the first picture to the right. Assume C is well-connected to the network. A sends a ping to B and B broadcasts it to C. Now before A has received any replies to that ping, a new host D connects to B and sends a ping. Because B's cache is still empty, B must broadcast D's ping if it wishes to respond immediately.



This second broadcast can be avoided through a technique we call *ping multiplexing*. The basic idea is to "multiplex" many incoming pings into a single outgoing ping per connection. Conversely we "demultiplex" a single incoming pong into multiple outgoing pongs. In the above example, B does not broadcast D's ping. Instead it waits until it has received pongs from C and then send these along to both A and D. This is illustrated in the last second and third pictures to the right. For the sake of simplicity, this illustration ignores the forwarded ping sent from B to A in response to D's ping.

To summarize: respond to a ping immediately with cached pongs, then set up information so that later pongs will be routed to the connection as needed. This can be implemented by maintaining two additional pieces of information for each connection: the GUID of the last ping received from that connection and how many pongs the connection still needs. The information is used as follows:

- When receiving a ping from a connection, store the GUID in the connection. Respond immediately with as many cached pongs as possible, but not more than N. Set the number of pongs the connection still needs appropriately.
- When receiving a pong from a connection, forward the pong along all connections still needing more pongs, using the GUIDs stored on those connections.

With this implementation, traditional ping routing tables are not needed!

Dealing with Old Clients

If all clients used the above scheme, at most one ping and M pongs would be sent per connection per T seconds. Unfortunately, new clients will have to deal with old and malicious clients that forward pings at unbounded speeds. For this reason, our scheme limits the rate that incoming pings are accepted; any pings exceeding that rate are dropped. This is implemented by maintaining the time that the last ping was accepted from a connection. Again, we stress that this is solely to defend against old clients and malicious attackers.

It is also wise to treat older clients specially when sending pings and receiving pongs. First, a new client should ping an old client no more than once every 30 seconds or so. The reason is that pings to old clients result in true broadcasts and are harmful to the network. Moreover, older clients will likely return a flood of pongs, consuming a lot of the new client's bandwidth. Second, pongs from older clients should be placed in a special reserve cache. Because these pongs include hosts that cannot accept incoming connections, they should only be used when absolutely necessary. For example, if a host lost all connections shortly after the normal cache expired, the reserve cache could be used to establish new connections.

Luckily we have a way to identify older clients. As discussed at the O'Reilly Conference and on the GDF, clients should handshake by initially sending pings with a marked GUID. The ninth byte of this GUID (i.e., GUID[8] in C or Java) is 0xFF, and the last byte is the protocol version number. (Note that this ping handshaking has nothing to do with the "GNUTELLA CONNECT 0.4" handshake string sent; unfortunately this string must be frozen for the sake of backwards compatibility.) The current protocol version number is 0, which stands for "Gnutella 0.5". We propose to use a version number of 1 for our ping/pong scheme and call this "Gnutella 0.6".

Pseudocode

Below is pseudocode that puts all of the above ideas together. In our pseudocode, cache refreshes are triggered when receiving a ping T seconds after the last cache refill. Another implementation is to use a timer to refresh the cache exactly every T seconds. In practice, both will behave nearly the same.

There are some special cases below for dealing with old clients. However, we have omitted some details, such as sending periodic update pings to old clients and using the reserve cache when no new pongs can be found. Also, there is a special case for sending handshake pings and receiving pings from a network crawler.

Constants:

- T: both the expire time for cached pongs and the throttle time. Suggested value: 3 seconds.
- M: the max TTL. Suggested value: 7.
- N: the max number of pongs to return in response to a ping. Suggested value: 10.

Global variables:

- CACHE: the pong cache. An M array list, each containing a list of <address, connection> pairs. Associating an address with a connection prevents us from sending a cached pong down the path it came from.
- NEXT_EXPIRE: the time we will next expire CACHE. At this time, we will also forward pings to all connections.
- RESERVE_CACHE: the cache of expired pongs and pongs from old clients. Used in case the cache empties and all connections suddenly die.

Maintain the following variables per connection C:

- ACCEPT_TIME the time we will next *accept* a ping from C.
- ACCEPT_GUID: the GUID of the last ping accepted on C, or null if not seen.
- NEEDED: an array of size M. NEEDED[i] is the number of pongs with hops value i needed by C.

When establishing a new connection C:

- If you don't need any connections and have enough values in RESERVE_CACHE, send C a new ping with TTL 1 for handshaking purposes.
- Otherwise send C a new ping with TTL=M.

When receiving a ping P on connection C, even if C is a "temporary/reject connection":

- Throttle incoming pings to defend against old clients and attackers. If the current time is less than C.ACCEPT_TIME, drop P and return from this method. Otherwise set C.ACCEPT_TIME to be the current time+T.
- Refresh the cache if necessary, being sure to synchronize properly. If the current time is greater than NEXT_EXPIRE, do the following:
 - Set NEXT_EXPIRE=current time+N.
 - Empty the cache, copying the contents into RESERVE_CACHE.
 - Send a ping with a new GUID and TTL M to all connections to newer clients. *[sic]*
- Special case for crawlers: if P.TTL=2 and P.HOPS=0, send pongs with the address and ports of all neighbors. Return from this method.
- Setup pong demultiplexing tables. Set C.ACCEPT_GUID=P.GUID. For all $0 < i \leq P.TTL$, set C.NEEDED[i]=N/P.TTL. For all $i > P.TTL$, set C.NEEDED[i]=0.
- Return a pong with my address if I can accept incoming connections right now.
- Return cached pongs. For all i between 1 and P.TTL, inclusive,
 - Choose NEEDED[i] addresses from CACHE[i] as possible, or as many as possible. Be sure that no address was received from C.
 - Decrement C.NEEDED[i] by the amount chosen.
 - Send the addresses to C in pongs with P.GUID.

When receiving a pong P on connection C:

- If C is an old client, place in RESERVE_CACHE and return from this method.

- Add <P.ADDRESS, C> to CACHE[P.HOPS].
- Route/demultiplex P. For each connection C' except for C,
 - Check if C' still needs a pong. If C'.NEEDED[P.HOPS]==0, continue to the next connection.
 - Send P along C' with GUID given by C'.ACCEPT_GUID.
 - Decrement C'.NEEDED[P.HOPS]

Appendix: Estimating Horizon Size

Our ping/pong scheme does not provide a way to measure the horizon size. We do acknowledge, however, that there is some value in providing reasonable horizon statistics. This information tells the user whether she is well-connected to the network and may be useful in helping a client establish better connections. Here we outline some ways of measuring the horizon size.

The easiest way is to maintain a large set of all pongs seen, including those from older connections. Eventually this set will contain pongs from most non-firewalled hosts in the horizon. However, this set will grow very slowly, since at most 10 pongs are added per 3 seconds per connection. Furthermore, the client should take care to expire entries that are more than a few minutes old.

A similar approach is to maintain a set of all hosts responding to queries. (Remember that the query reply message includes the address of the host sharing the file.) This also has some problems. First, this scheme only measures those hosts in the horizon that are sharing files, although this is arguably as important as the total horizon size. Secondly this set will also grow slowly since query replies only consume a small percentage of all messages. Finally, it is difficult to measure the number of files being shared without adding extra information to the query hit descriptor.

The best approach is to introduce one or two messages solely for the purpose of communicating horizon size. Assume for simplicity that the Gnutella network is a tree, i.e., there is only one path between any two hosts. A key observation is that the number of hosts reachable from some host with TTL N is the sum of the number of hosts reachable from all its neighbors with TTL N-1. For formally, let $H[A, n, B]$ be the number of hosts within n hops of host A that are reachable through B. Let $H'[A, n, B]$ be the number of hosts within n hops of host A that are *not* reachable through B. Then for any host A with neighbors $N_1 \dots N_m$,

$$\begin{aligned} H[A, n, N_i] &= 1 + H'[N_i, n-1, A] \\ H'[A, n, N_i] &= H[A, n, N_1] + \dots + H[A, n, N_{i-1}] + H[A, n, N_{i+1}] + \dots + H[A, n, N_m] \\ H'[A, 0, N_i] &= 0 \end{aligned}$$

Horizon estimation now works through a sort of dynamic programming algorithm. Each host maintains the horizon size per connection, i.e., $H[A, n, N_i]$ for all n and i . Every few seconds or so, hosts calculate their H' tables from these values, exchange them with neighbors, and use these values to update their H tables. The total horizon size reported to the user is the sum over all i of $H[A, n, N_i]$. This scheme is accurate, efficient, and uses almost no bandwidth.

Search

GO! >>

Advanced Search >>

news & info

solutions

international

careers


bell labs

site info

imminet

Lucent Technologies

Bell Labs Innovations



solutions

communications revolution

full circle

internetworking

optical networking

switching

wireless

software

network products

network care services

technical support

contact us


press room


analyst room

SPEED IS EVERYTHING

Introducing **imminet**... Great content delivery solutions from Lucent Technologies. Now, you can take advantage of a robust portfolio of **imminet** products that help enable smarter and faster networks. This lets your happy customers experience the shortest possible distance between them and the content they need.

The suite of **imminet** products mean you'll provide your customers with an intelligent network for content delivery and distribution that boosts network performance and reduces delays. And products are built around the **imminet** Dynamic Architecture. So, accelerate your network by asking for **imminet** today.

 [Click Here for more information](#)

 [Click Here for more information](#)


The scalable carrier-grade **imminet** WebCache helps to eliminate access latency, deliver fresh content, and reduce server overload. The WebCache 50, 100 and 200 models are self-contained content and application servers powered by Lucent's cache-optimized software. They proactively retrieve, replicate and store Web content at a server location close to the user.


[Learn more.](#)


The **imminet** WebDirector platform combines high-speed flow setup with wire-speed switching and forwarding performance to improve web access speed and response time. The WebDirector offers smart traffic switching based on criteria ranging from IP address to URLs. The WebDirector 80 and 180 models optimize Web response time and content availability by sending content requests to the server or WebCache best able to handle the request. Load balancing is done across many devices including servers, caches and firewalls.


[Learn more.](#)

Documentation

 [imminet Content Solutions Brochure](#)

 [Caching Techniques for Streaming Multimedia over the Internet](#)

 [Streaming and Broadcasting over the Internet](#)

 [Design and Implementation of a Caching System for Streaming Media over the Internet](#)

myLucent

Your personalized portal to the world of Lucent online

[click here](#)

Lucent Home

Service Providers

Agere Systems/Microelectronics

Worldwide Services

Investor Relations

Copyright © 2000-2001 Lucent Technologies, Inc. All Rights Reserved. Use of this site indicates you accept the [Terms of Use](#) and the [Privacy Statement](#). For comments or questions about this site, [contact us](#).

162

Microsoft Exhibit 1022

<https://web.archive.org/web/20010124044000/http://www.lucent.com/serviceprovider/imminet/>



Lir

[about](#) [download](#) [support](#) [resources](#) [news](#) [contact](#)

Baseline Healthy Client Requirements

Improving the Gnutella Network

We wrote up some of our thoughts on what makes for healthy behaviour of a Gnutella servent. It is more focused on medium term changes but perhaps worth a read. <http://www.limewire.com/health.htm>

Let me go through what we think the baseline for a healthy client NOW should be.

If we were to remove Gnutella 0.56 clients and get other clients to update themselves, what should they do to promote health on the network?

I'll go into more details but the brief answer would be:

- 1) Have an incomplete directory and automatically share downloads.
- 2) Block browser uploads.
- 3) Avoid network broadcasts.
- 4) Implement some form of network rebalancing.

The other way to say this is:

- 1) Encourage sharing of content.
- 2) Reduce freeloaders.
- 3) Reduce unnecessary network traffic.
- 4) Encourage a healthy network structure.

[top](#)

1.0

Point one is fairly obvious and we have been discussing that in the forum a little. Note that you should have an incomplete download directory to prevent the sharing of partial files but then the completed downloads should be shared by default.

[top](#)

2.0

Now blocking browsers is regrettable and not necessarily what we want to do in the long term. However, for the short term, they are flooding the Gnutella network with upload requests and giving the active client users a bad experience. They take up most of the upload slots on the network without sharing. You can detect the majority of browser uploads by looking for "Mozilla" in the User-Agent header string. I can give a list of download accelerators that we block as well if interested. In fact, LimeWire and BearShare send back an HTML page (or redirect) encouraging users to join the network and share.

[top](#)

3.0

Our paper outlines some future idea of avoiding network broadcasts. The most important right now is to not broadcast pushes. We have also been discussing that in the forum. Broadcasting in general should be kept to a minimum - i.e. don't actively ping the network.

[top](#)

4.0

So, I think the network rebalancing needs some mention. We currently have a much healthier network than a few months ago but as the network grows, this could become more of an issue again. The basic idea is that if all servents behave in a particular fashion, the network can heal itself and remove some of the congestion due to the modem bandwidth barrier. The

EXHIBIT A

simple answer here is to drop a connection if it gets clogged. What we hope that this will cause is for the slower modem users to move out from the center or fast areas of the network. Obviously, encouraging the use of reflector is good - if there are publicly accessible reflectors. Here are our rules for dropping a connection.

Drop clients if:

- +They have been unable to receive our data for ~10 seconds.

- +They have sent us no data for ~10 seconds AND do not respond to a ping with TTL=1 AND are not a Clip2 reflector.

One other thing that we would recommend is message prioritization on sending. If you have a certain I/O limitation, you might need to drop some messages in a smart fashion. Obviously, you prioritize your own query messages first but those are a tiny part of your traffic.

The best order for prioritization is this:

- Pushes

- Query Replies (Hits)

- Queries

- Pongs

- Pings

If you need to drop packets, drop from pings and pongs on up. This will help pushes work and allow the network to function reasonably under stress. Note though, if there is an overly slow or clogged connection, it needs to be dropped. Perhaps this qualifies as brute force flow control. The basic goal here is that one slow connection should not prevent others from making progress.

There are likely other improvements that we can all think of. (multi-direction push routing?) However, I believe that this set would create a much better network.

[top](#)

[home](#) | [about](#) | [download](#) | [support](#) | [resources](#) | [press](#) | [contact](#) | [legal](#) | [sitemap](#)

Freenet: A Distributed Anonymous Information Storage and Retrieval System

Ian Clarke
124C Lyham Road, Clapham Park
London SW2 5QA
United Kingdom
i.clarke@dynamicblue.com

Oskar Sandberg
Mörbydalen 12
18252 Stockholm
Sweden
md98-osa@nada.kth.se

Brandon Wiley
2305 Rio Grande St.
Austin, TX 78705
USA
blanu@uts.cc.utexas.edu

Theodore W. Hong *
Department of Computing
Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ
United Kingdom
t.hong@doc.ic.ac.uk

July 1, 2000

Abstract

We describe Freenet, a peer-to-peer network application that permits the publication, replication, and retrieval of data while protecting the anonymity of both authors and readers. Freenet operates as a network of identical nodes that collectively pool their storage space to store data files, and cooperate to route requests to the most likely physical location of data. No broadcast search or centralized location index is employed. Files are referred to in a location-independent manner, and are dynamically replicated in locations near requestors and deleted from locations where there is no interest. It is infeasible to discover the true origin or destination of a file passing through the network, and difficult for a node operator to determine or be held responsible for the actual physical contents of her own node.

1 Introduction

Computer networks are rapidly growing in importance as a medium for the storage and exchange of information. However, current systems afford little privacy to their users, and typically store any given data item in only one or a few fixed places, creating a central point of failure. Because of a continued desire among individuals to protect the privacy of their authorship or readership of various types of sensitive information[24], and the undesirability of central points of failure

*Supported by grants from the Marshall Aid Commemoration Commission and the National Science Foundation.

which can be attacked by opponents wishing to remove data from the system[10, 23] or simply overloaded by too much interest[1], systems offering greater security and reliability are needed.

We are developing Freenet, a distributed information storage and retrieval system designed to address these concerns of privacy and availability. The system operates as a location-independent distributed file system across many individual computers that allows files to be inserted, stored, and requested anonymously. There are five main design goals:

- Anonymity for both producers and consumers of information
- Deniability for storers of information
- Resistance to attempts by third parties to deny access to information
- Efficient dynamic storage and routing of information
- Decentralization of all network functions

The system is designed to respond adaptively to usage patterns, transparently moving, replicating, and deleting files as necessary to provide efficient service without resorting to broadcast searches or centralized location indexes. It is not intended to guarantee permanent file storage, although it is hoped that enough nodes will join with enough storage capacity that most files will be able to remain indefinitely. In addition, the system operates at the application layer and assumes the existence of a secure transport layer, although it is transport-independent. It does not seek to provide anonymity for general network usage, only for Freenet file transactions.

Freenet is currently being developed as a free software project on Sourceforge, and a preliminary implementation can be downloaded from <http://freenet.sourceforge.net/>. It grew out of work originally done by the first author at the University of Edinburgh[11].

2 Related work

Several strands of related work in this area can be distinguished. Anonymous point-to-point channels based on Chaum's mix-net scheme[7] have been implemented for email by the Mix-master remailer[19] and for general TCP/IP traffic by onion routing[17] and Freedom[27]. Such channels are not in themselves easily suited to one-to-many publication, however, and are best viewed as a complement to Freenet since they do not provide file access and storage.

Anonymity for consumers of information in the web context is provided by browser proxy services such as the Anonymizer[5], although they provide no protection for producers of information and do not protect consumers against logs kept by the services themselves. Private information retrieval schemes[9] provide much stronger guarantees for information consumers, but only to the extent of hiding which piece of information was retrieved from a particular server. In most cases, the fact of contacting a particular server in itself reveals much about the information retrieved, which can only be counteracted by having each server hold all information (naturally this scales poorly). The closest work to our own is Reiter and Rubin's Crowds system[21], which uses a similar method of proxying requests for consumers, although Crowds does not itself store information and does not protect information producers. Berthold

et al. propose Web Mixes[6], a stronger system which uses message padding and reordering and dummy messages to increase security, but again does not protect information producers.

The Rewebber[22] provides a measure of anonymity for producers of web information by means of an encrypted URL service which is essentially the inverse of an anonymizing browser proxy, but has the same difficulty of providing no protection against the operator of the service itself. TAZ[16] extends this idea by using chains of nested encrypted URLs which successively point to different rewebber servers to be contacted, although this is vulnerable to traffic analysis using replay. Both rely on a single server as the ultimate source of information. Publius[26] enhances availability by distributing files as redundant shares among n web servers, only k of which are needed to reconstruct a file; however, since the identity of the servers themselves is not anonymized, an attacker might remove information by forcing the closure of $n-k+1$ servers. The Eternity proposal[4] seeks to archive information permanently and anonymously, although it lacks specifics on how to efficiently locate stored files, making it more akin to an anonymous backup service. Free Haven[12] is an interesting anonymous publication system which uses a trust network and file trading mechanism to provide greater server accountability while maintaining anonymity.

distributed.net[13] demonstrated the concept of pooling computer resources among multiple users on a large scale for CPU cycles; other systems which do the same for disk space are Napster[20] and Gnutella[15], although the former relies on a central server to locate files and the latter employs an inefficient broadcast search. Neither one replicates files. Intermemory[8] and India[14] are cooperative distributed filesystem systems intended for long-term archival storage along the lines of Eternity, in which files are split into redundant shares and distributed among many participants. Akamai[2] provides a service which replicates files at locations near information consumers, but is not suitable for producers who are individuals (as opposed to corporations). None of these systems attempt to provide anonymity.

3 Architecture

Freenet is implemented as a peer-to-peer network of nodes that query one another to store and retrieve data files, which are named by location-independent keys. Each node maintains its own local datastore which it makes available to the network for reading and writing, as well as a dynamic routing table containing addresses of other nodes and the keys that they are thought to hold. It is intended that most users of the system will run nodes, both to provide security guarantees against inadvertently using a hostile foreign node and to increase the storage capacity available to the network as a whole.

The system can be regarded as a cooperative distributed filesystem incorporating location independence and transparent lazy replication. Just as systems such as **distributed.net**[13] enable ordinary users to share unused CPU cycles on their machines, Freenet enables users to share unused disk space. However, where **distributed.net** uses those CPU cycles for its own purposes, Freenet is directly useful to users themselves, acting as an extension to their own hard drives.

The basic model is that queries are passed along from node to node in a chain of proxy requests, with each node making a local routing decision in the style of IP routing about where

to send the query next (this varies from query to query). Nodes know only their immediate upstream and downstream neighbors in the chain. Each query is given a “hops-to-live” count which is decremented at each node to prevent infinite chains (analogous to IP’s time-to-live). Each query is also assigned a pseudo-unique random identifier, so that nodes can prevent loops by rejecting queries they have seen before. When this happens, the immediately preceding node simply chooses a different node to forward to. This process continues until the query is either satisfied or exceeds its hops-to-live limit. Then, the success or failure result is passed back up the chain to the sending node.

No node is privileged over any other node, so no hierarchy or central point of failure exists. Joining the network is simply a matter of discovering the address of one or more existing nodes through out-of-band means, then starting to send messages. Files are immutable at present, although file updatability is on the agenda for future releases. In addition, the namespace is currently flat, consisting of the space of 160 bit SHA-1[3] hashes of descriptive text strings, although support for a richer namespace is a priority for development. See section 5 for further discussion of updating and naming.

3.1 Retrieving data

To retrieve data, a user hashes a short descriptive string (for example, `text/philosophy/sun-tzu/art-of-war`) to obtain a file key. (See section 5 for details on how descriptive strings corresponding to files can be discovered.) She then sends a request message to her own node specifying that key and a hops-to-live value. When a node receives a request, it first checks its own store for the data and returns it if found, together with a note saying it was the source of the data. If not found, it looks up the nearest key in its routing table to the key requested and forwards the request to the corresponding node. If that request is ultimately successful and returns with the data, the node will pass the data back to the upstream requestor, cache the file in its own datastore, and create a new entry in its routing table associating the actual data source with the requested key. A subsequent request for the same key will be immediately satisfied from the local cache; a request for a “similar” key (determined by lexicographic distance) will be forwarded to the previously successful data source. Because maintaining a table of data sources is a potential security concern, any node along the way can unilaterally decide to change the reply message to claim itself or another arbitrarily-chosen node as the data source.

If a node cannot forward a request to its preferred downstream node because the target is down or a loop would be created, the node having the second-nearest key will be tried, then the third-nearest, and so on. If a node runs out of candidates to try, it reports failure back to its upstream neighbor, which will then try *its* second choice, etc. In this way, a request operates as a steepest-ascent hill-climbing search with backtracking. If the hops-to-live count is exceeded, a failure result is propagated back to the original requestor without any further nodes being tried. Nodes may unilaterally curtail excessive hops-to-live values to reduce network load. They may also forget about pending requests after a period of time to keep message memory free.

Figure 1 depicts a typical sequence of request messages. The user initiates a request at node *a*. Node *a* forwards the request to node *b*, which forwards it to node *c*. Node *c* is unable to contact any other nodes and returns a backtracking “request failed” message to *b*. Node *b* then tries its second choice, *e*, which forwards the request to *f*. Node *f* forwards the request to *b*,

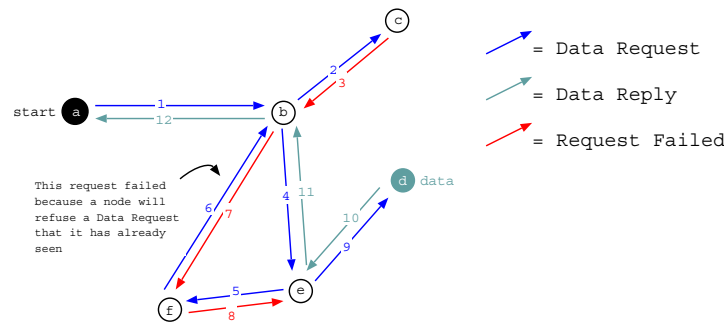


Figure 1: A typical request sequence.

which detects the loop and returns a backtracking failure message. Node *f* is unable to contact any other nodes and backtracks one step further back to *e*. Node *e* forwards the request to its second choice, *d*, which has the data. The data is returned from *d* via *e* and *b* back to *a*, which sends it back to the user. The data is also cached on *e*, *b*, and *a*.

This mechanism has a number of effects. Most importantly, we hypothesize that the quality of the routing should improve over time, for two reasons. First, nodes should come to specialize in locating sets of similar keys. If a node is listed in routing tables under a particular key, it will tend to receive mostly requests for keys similar to that key. It is therefore likely to gain more “experience” in answering those queries and become better informed in its routing tables about which other nodes carry those keys. Second, nodes should become similarly specialized in storing clusters of files having similar keys. Because forwarding a request successfully will result in the node itself gaining a copy of the requested file, and most requests will be for similar keys, the node will mostly acquire files with similar keys. Taken together, these two effects should improve the efficiency of future requests in a self-reinforcing cycle, as nodes build up routing tables and datastores focusing on particular sets of keys, which will be precisely those keys that they are asked about.

In addition, the request mechanism will cause popular data to be transparently replicated by the system and mirrored closer to requestors. For example, if a file that is originally located in London is requested in Berkeley, it will become cached locally and provide faster response to subsequent Berkeley requests. It also becomes copied onto each computer along the way, providing redundancy if the London node fails or is shut down. (Note that “along the way” is determined by key closeness and does not necessarily have geographical relevance.)

Finally, as nodes process requests, they create new routing table entries for previously-unknown nodes that supply files, increasing connectivity. This helps new nodes to discover more of the network (although it does not help the rest of the network to discover *them*; for that, performing inserts is necessary). Note that direct links are created, bypassing the intermediate nodes used. Thus, nodes that successfully supply data will gain routing table entries and be contacted more often than nodes that do not.

Because hashes are used as keys, lexicographic closeness of keys does not imply any closeness of the original descriptive strings and presumably, no closeness of subject matter of the files.

This lack of semantic closeness is not important, however, as the routing algorithm is based on knowing where keys are located, not where subjects are located. That is, supposing `text/philosophy/sun-tzu/art-of-war` hashes to AH5JK2, requests for this file can be routed more effectively by creating clusters containing AH5JK1, AH5JK2, and AH5JK3, not by creating clusters for works of philosophy. Indeed, the use of a hash is desirable precisely because philosophical works will be scattered across the network, lessening the chances that failure of a single node will make all philosophy unavailable.

3.2 Storing data

Inserts follow a parallel strategy to requests. To insert data, a user picks an appropriate descriptive text string and hashes it to create a file key. She then sends an insert message to her own node specifying the proposed key and a hops-to-live value (this will determine the number of nodes to store it on). When a node receives an insert proposal, it first checks its own store to see if the key is already taken. If the key is found, the node returns the pre-existing file as if a request had been made for it. The user will thus know that a collision was encountered and can try again using a different key, i.e. different descriptive text. (Although potentially the use of a hash might cause extra collisions, in practice a high-quality hash of sufficient length should only cause collisions if the same initial descriptive text was chosen.) If the key is not found, the node looks up the nearest key in its routing table to the key proposed and forwards the insert to the corresponding node. If that insert causes a collision and returns with the data, the node will pass the data back to the upstream inserter and again behave as if a request had been made (i.e. cache the file locally and create a routing table entry for the data source).

If the hops-to-live limit is reached without a key collision being detected, an “all clear” result will be propagated back to the original inserter. Note that for inserts, this is a successful result, in contrast to the request case. The user then sends the data to insert, which will be propagated along the path established by the initial query and stored in each node along the way. Each node will also create an entry in its routing table associating the inserter (as the data source) with the new key. To avoid the obvious security problem, any node along the way can unilaterally decide to change the insert message to claim itself or another arbitrarily-chosen node as the data source.

If a node cannot forward an insert to its preferred downstream node because the target is down or a loop would be created, the insert backtracks to the second-nearest key, then the third-nearest, and so on in the same way as for requests. If the backtracking returns all the way back to the original inserter, it indicates that fewer nodes than asked for could be contacted. As with requests, nodes may curtail excessive hops-to-live values and/or forget about pending inserts after a period of time.

This mechanism has three effects. First, newly inserted files are selectively placed on nodes already possessing files with similar keys. This reinforces the clustering of keys set up by the request mechanism. Second, new nodes can tell the rest of the network of their existence by inserting data. Third, an attempt by an attacker to supplant an existing file by deliberate insert collision (e.g., by inserting a corrupted or empty file under the same key) is likely to simply spread the real file further, since the original file is propagated back on collision. (The use of a content-hash key, described in section 5, makes such an attack infeasible altogether.)

3.3 Managing data

All information storage systems must deal with the problem of finite storage capacity. Individual Freenet node operators can configure the amount of storage to dedicate to their datastores. Node storage is managed as an LRU (Least Recently Used) cache[25], with data items kept sorted in decreasing order by time of most recent request (or time of insert, if an item has never been requested). When a new file arrives (from either a new insert or a successful request) which would cause the datastore to exceed the designated size, the least recently used files are evicted in order until there is room. The impact on availability is mitigated somewhat by the fact that the routing table entries created when the evicted files first arrived will remain for a time, potentially allowing the node to later get new copies from the original data sources. (Routing table entries are also eventually deleted in a similar fashion as the table fills up, although they will be retained longer since they are smaller.)

Strictly speaking, the datastore is not a cache, since the set of datastores is all the storage that there is, i.e. there is no “permanent” copy which is being duplicated. Once all the nodes have decided, collectively speaking, to drop a particular file, it will no longer be available to the network. In this respect, Freenet differs from the Eternity service, which seeks to provide guarantees of file lifetime.

This mechanism has an advantageous side, however, since it allows outdated documents to fade away after being superseded by newer documents, thus alleviating some of the problem of immutable files. If an outdated document is still used and considered valuable for historical reasons, it will stay alive precisely as long as it continues to be requested.

For political or legal reasons, it may be desirable for node operators not to explicitly know the contents of their datastores. Therefore, it is recommended that all inserted files be encrypted by their original unhashed descriptive text strings in order to obscure their contents. Of course, this does not secure the file—that would be impossible since a requestor (potentially anyone) must be capable of decrypting the file once retrieved. Rather, the objective is that the node operator can plausibly deny any knowledge of the contents of her datastore, since all she knows *a priori* is the hashed key and its associated encrypted file. The hash cannot feasibly be reversed to reveal the unhashed description and decrypt the file. With effort, of course, a dictionary attack will reveal which keys are present—as it must in order for requests to work at all.

4 Protocol details

The Freenet protocol is packet-oriented and uses self-contained messages. Each message includes a transaction ID so that nodes can track the state of inserts and requests. This design is intended to permit flexibility in the choice of transport mechanisms for messages, whether they be TCP, UDP, or other technologies such as packet radio. For efficiency, nodes are also able to send multiple messages over a persistent channel such as a TCP connection, if available. Node addresses consist of a transport method plus a transport-specific identifier (such as an IP address and port number), e.g. `tcp/192.168.1.1:19114`.

A Freenet transaction begins with a Request.Handshake message from one node to another,

specifying the desired return address of the sending¹ node. (The return address may be impossible to determine from the transport layer alone, or may use a different transport from that used to send the message.) If the remote node is active and responding to requests, it will reply with a Reply.Handshake specifying the protocol version number that it understands. Handshakes are remembered for a few hours, and subsequent transactions between the same nodes during this time may omit this step.

All messages contain a randomly-generated 64-bit transaction ID, a hops-to-live counter, and a depth counter. Although the ID cannot be guaranteed to be unique, the likelihood of a collision occurring during the transaction lifetime among the limited set of nodes that it sees is extremely low. Hops-to-live is set by the originator of a message and is decremented at each hop to prevent messages being forwarded indefinitely. To reduce the information that an attacker can obtain from the hops-to-live value, messages do not automatically terminate after hops-to-live reaches 1 but are forwarded on with finite probability (with hops-to-live again 1). Depth is incremented at each hop and is used by a replying node to set hops-to-live high enough to reach a requestor. Requestors should initialize it to a small random value to obscure their location. As with hops-to-live, a depth of 1 is not automatically incremented but is passed unchanged with finite probability.

To request data, the sending node sends a Request.Data message specifying a transaction ID, initial hops-to-live and depth, and a search key. The remote node will check its datastore for the key and if not found, will forward the request to another node as described in section 3.1. Using the chosen hops-to-live count, the sending node starts a timer for the expected amount of time it should take to contact that many nodes, after which it will assume failure. While the request is being processed, the remote node may periodically send back Reply.Restart messages indicating that messages were stalled waiting on network timeouts, so that the sending node knows to extend its timer.

If the request is ultimately successful, the remote node will reply with a Send.Data message containing the data requested and the address of the node which supplied it (possibly faked). If the request is ultimately unsuccessful and its hops-to-live are completely used up trying to satisfy it, the remote node will reply with a Reply.NotFound. The sending node will then decrement the hops-to-live of the Send.Data (or Reply.NotFound) and pass it along upstream, unless it is the actual originator of the request. Both of these messages terminate the transaction and release any resources held. However, if there are still hops-to-live remaining, usually because the request ran into a dead end where no viable non-looping paths could be found, the remote node will reply with a Request.Continue giving the number of hops-to-live left. The sending node will then try to contact the next-most likely node from its routing table. It will also send a Reply.Restart upstream.

To insert data, the sending node sends a Request.Insert message specifying a randomly-generated transaction ID, an initial hops-to-live and depth, and a proposed key. The remote node will check its datastore for the key and if not found, forward the insert to another node as described in section 3.2. Timers and Reply.Restart messages are also used in the same way as for requests.

If the insert ultimately results in a key collision, the remote node will reply with either

¹Note that the sending node may not be the original requestor.

a `Send.Data` message containing the existing data or a `Reply.NotFound` (if existing data was not actually found, but routing table references to it were). If the insert does not encounter a collision, yet runs out of nodes with nonzero hops-to-live remaining, the remote node will reply with a `Request.Continue`. In this case, `Request.Continue` is a failure result meaning that not as many nodes could be contacted as asked for. These messages will be passed along upstream as in the request case. Both messages terminate the transaction and release any resources held. However, if the insert expires without encountering a collision, the remote node will reply with a `Reply.Insert`, indicating that the insert can go ahead. The sending node will pass along the `Reply.Insert` upstream and wait for its predecessor to send a `Send.Insert` containing the data. When it receives the data, it will store it locally and forward the `Send.Insert` downstream, concluding the transaction.

5 Naming, searching, and updating

Freenet's basic flat namespace has obvious disadvantages in terms of discovering documents, name collisions, etc. Several mechanisms of providing more structure within the current scheme are possible. For example, directory-like documents containing hypertext pointers to other files could be created. A directory file under the key `text/philosophy` could contain a list of keys such as `text/philosophy/sun-tzu/art-of-war`, `text/philosophy/confucius/analects`, and `text/philosophy/nozick/anarchy-state-utopia`, using appropriate syntax interpretable by a client. The difficulty, however, is in deciding how to permit changes to the directory to update entries, while preventing the directory from being corrupted or spammed. An alternative mechanism is to encourage individuals to maintain and share their own compilations of keys—subjective bookmark lists, rather than authoritative directories. This is the approach in common use on the world-wide web.

Name collisions are still a problem with both bookmark lists and directories. One way of addressing collisions is to introduce a two-level structure similar to that used by most traditional file systems. Real files could be stored under a pseudo-unique binary key, such as a hash of the files' contents (a *content-hash key*). Users would access these files by first retrieving an indirect file stored under a semantically meaningful name. The indirect file would consist solely of a list of binary keys corresponding to that name (possibly only one), along with other information useful for differentiating among the possible choices, such as author, creation time, and endorsements by other users. Content-hash keys would also protect against files being maliciously tampered with or replaced. However, indirect files are essentially like low-level directories, and share the same problem of managing updates. Another approach is to skip the indirect files altogether and have bookmark lists pointing to content-hash keys, rather than names.

Introducing a search capability in conjunction with binary keys offers a way to side-step the need to maintain directories. The most straightforward way to add search is to run a hypertext spider such as those used to search the web. While an attractive solution in many ways, this conflicts with the design goal of avoiding centralization. A possible alternative is to create a special class of lightweight indirect files. When a real file is inserted, the author could also insert a number of indirect files containing a single pointer to the real file, named according to search keywords chosen by her. These indirect files would differ from normal files in that collisions

would be permitted on insert, and requests for an indirect file key (i.e. a keyword) would keep going until a specified number of indirect files (i.e. search results) were accumulated. Managing the likely large volume of these indirect files is an open problem.

Updates by a single user can be handled in a reasonably straightforward manner by using a variation on indirection. When an author inserts a file which she later intends to update, she first generates a public-private key pair and signs the file with the private key. The file is inserted under a binary key, but instead of using the hash of the file's contents, the literal public key itself is used (a *signature-verifying key*). As with inserting under a content-hash key, inserting under a signature-verifying key provides a pseudo-unique binary key for a file, which can also be used to verify that the file's contents have not been tampered with. To update the file, the new version is signed by the private key and inserted under the public signature-verifying key. When the insert reaches a node which possesses the old version, a key collision will occur. The node will check the signature on the new version, verify that it is both valid and more recent, and replace the old version.

In order to allow old versions of files to remain in existence for historical reasons and to prevent the possibility of authors being compelled to "update" their own files out of existence, an additional level of indirection can be used. In this scheme, the real file is inserted under its content-hash key. Then, an indirect file containing a pointer to that content-hash key is created, but inserted under a signature-verifying key. This signature-verifying key is given out as the binary key for the file and entered, for example, in directories and bookmark lists (making them double indirect files). When the author creates a new version, it is inserted under its own content-hash key, distinct from the old version's key. The signature-verified indirect file is then updated to point to the new version (or possibly to keep pointers to all versions). Thus the signature-verifying key will always lead to the most recent version of the file, while old versions can continue to be accessed by content-hash key if desired. (If not requested, however, these old versions will eventually disappear like any other unused file.)

For large files, splitting files into multiple parts is desirable because of storage and bandwidth limitations. Splitting even medium files into standard-sized parts (e.g. 16 kilobytes) also has advantages in combating traffic analysis. This is easily accomplished by inserting each part separately under a content-hash key, and including pointers to the other parts.

Combining all these ideas together, a user might look through a bookmark list or perform a search on a keyword to get a list of signature-verifying binary keys for files dealing with a particular topic. Retrieving one of these keys gives an indirect file containing a set of content-hash keys corresponding to different versions, ordered by date. Retrieving the most recent content-hash key gives the first part of a multipart file, together with pointers to two other content-hash keys. Finally, retrieving those last two content-hash keys and concatenating the three parts together yields the desired file.

6 Performance simulation

Simulations were carried out on an early version of this system to give some indications about its performance. Here we summarize the most important results; for full details, see [11].

The scenario for these simulations was a network with between 500 and 900 nodes. Each

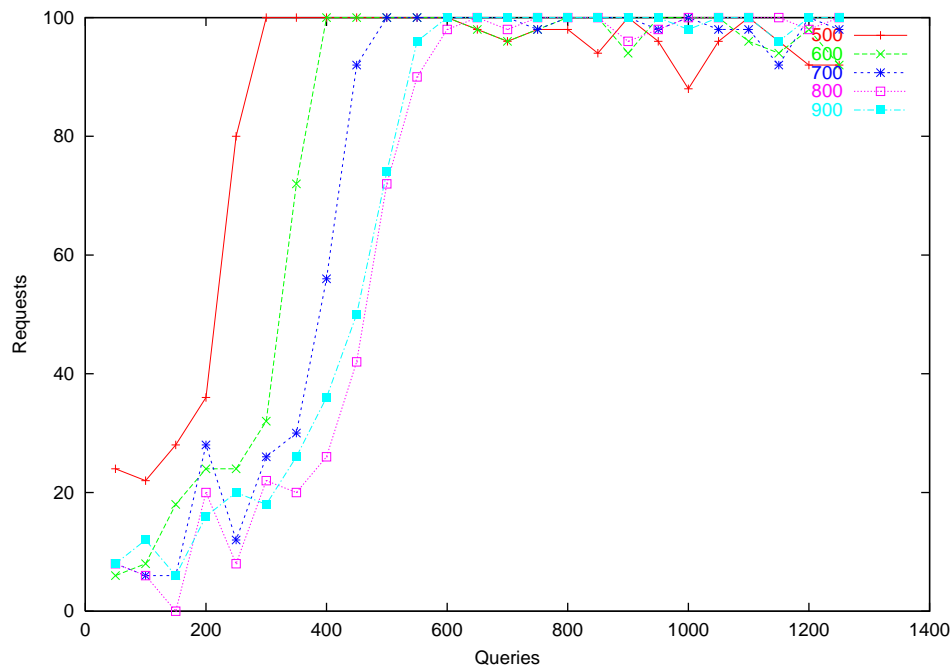


Figure 2: Percentage of successful requests over time.

node had a datastore size of 40 items and a routing table size of 50 addresses (relatively low, because of limitations on available hardware), and was given 10 unique items to store locally. The network was initially connected in a linear fashion, where each node started with routing references to one node on either side.

6.1 Retrieval success rate

Queries for random keys were sent to random nodes in the network, in batches of 50 parallel queries at a time, and the percentage of successful requests recorded over time. Figure 2 shows the percentage of successful requests versus the total number of queries since initialization, for several network sizes. We can see that the initially low success rate rises rapidly until over 95% of requests are successful. The number of queries until network convergence is approximately half the total size of the network.

6.2 Retrieval time

Queries for random keys were sent to random nodes in the network, in batches of 50 parallel queries at a time, and the average number of hops needed for a successful request recorded over time. Figure 3 shows the number of request hops versus the total number of queries since initialization, for several network sizes. We can see that the initially high number of hops

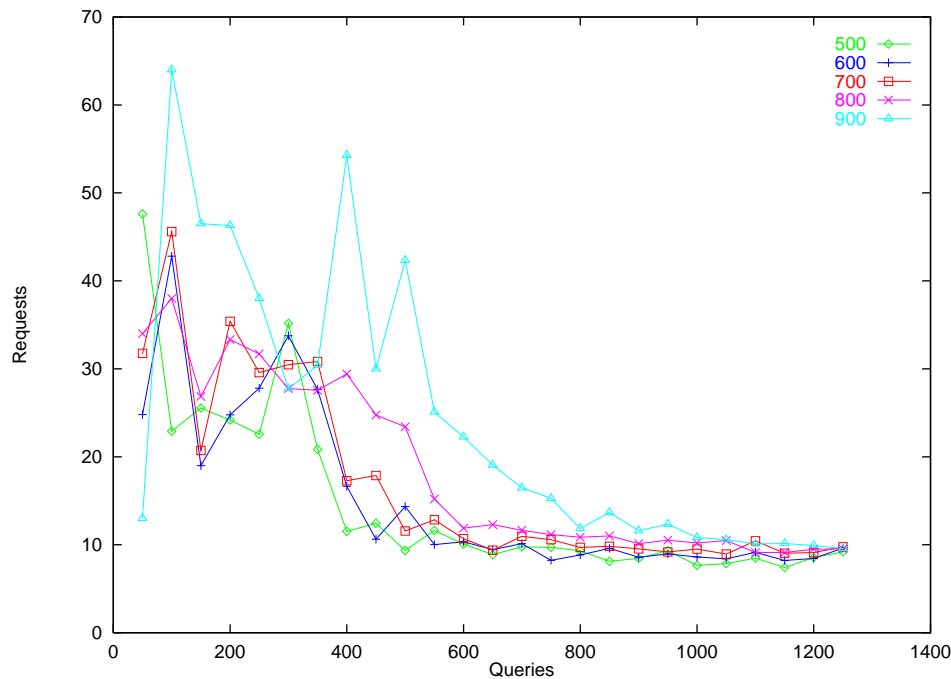


Figure 3: Number of hops per request over time.

required drops until it stabilizes at approximately 10 hops. This value changes remarkably little with network size, suggesting that the time required for requests should scale quite well. The number of queries until network convergence is approximately equal to the total size of the network.

7 Security

The primary goal for Freenet security is protecting the anonymity of requestors and inserters of files. It is also important to protect the identity of storer of files. Although trivially anyone can turn a node into a storer by requesting a file through it, thus “identifying” it as a storer, what is important is that there remain other, unidentified, holders of the file so that an adversary cannot remove a file by attacking all of the nodes that hold it. Files must be protected against malicious modification, and finally, the system must be resistant to denial-of-service attacks.

Reiter and Rubin[21] present a useful taxonomy of anonymous communication properties on three axes. The first axis is the type of anonymity: sender anonymity or receiver anonymity, which mean respectively that an adversary cannot determine either who originated a message, or to whom it was sent. The second axis is the adversary in question: a local eavesdropper, a malicious node or collaboration of malicious nodes, or a web server (not applicable to Freenet). The third axis is the degree of anonymity, which ranges from absolute privacy (the presence of

System	Attacker	Sender anonymity	Key anonymity
Basic Freenet	local eavesdropper	exposed	exposed
	collaborating nodes	beyond suspicion	exposed
Freenet + pre-routing	local eavesdropper	exposed	beyond suspicion
	collaborating nodes	beyond suspicion	exposed

Table 1: Anonymity properties of Freenet.

communication cannot be perceived) to beyond suspicion (the sender appears no more likely to have originated the message than any other potential sender), probable innocence (the sender is no more likely to be the originator than not), possible innocence, exposed, and provably exposed (the adversary can prove to others who the sender was).

As Freenet communication is not directed towards specific receivers, receiver anonymity is more accurately viewed as key anonymity, that is, hiding the key which is being requested or inserted. Unfortunately, since routing depends on knowledge of the key, key anonymity is not possible in the basic Freenet scheme (but see the discussion of “pre-routing” below). The use of hashes as keys provides a measure of obscurity against casual eavesdropping, but is of course vulnerable to a dictionary attack since unhashed keys must be widely known in order to be useful.

Freenet’s anonymity properties under this taxonomy are shown in Table 1. Against a collaboration of malicious nodes, sender anonymity is preserved beyond suspicion since a node in a request path cannot tell whether its predecessor in the path initiated the request or is merely forwarding it. [21] describes a probabilistic attack which might compromise sender anonymity, using a statistical analysis of the probability that a request arriving at a node a is forwarded on or handled directly, and the probability that a chooses a particular node b to forward to. This analysis is not immediately applicable to Freenet, however, since request paths are not constructed probabilistically. Forwarding depends on whether or not a has the requested data in its datastore, rather than chance. If a request is forwarded, the routing tables determine where it is sent to, and could be such that a forwards every request to b , or never forwards any requests to b , or anywhere in between. Nevertheless, the depth value may provide some indication as to how many hops away the originator was, although this is obscured by the random selection of an initial depth and the probabilistic means of incrementing it (see section 4). Similar considerations apply to hops-to-live. Further investigation is required to clarify these issues.

Against a local eavesdropper there is no protection on messages between the user and the first node contacted. Since the first node contacted can act as a local eavesdropper, it is recommended that the user only use a node on her own machine as the first point of entry into the Freenet network. Messages between nodes are encrypted against local eavesdropping, although traffic analysis may still be performed (e.g. an eavesdropper may observe a message going out without a previous message coming in and conclude that the target originated it).

Key anonymity and stronger sender anonymity can be achieved by adding mix-style “pre-routing” of messages. In this scheme, basic Freenet messages are encrypted by a succession of public keys which determine the route that the encrypted message will follow (overriding the

normal routing mechanism). Nodes along this portion of the route are unable to determine either the originator of the message or its contents (including the request key), as per the mix-net anonymity properties. When the message reaches the endpoint of the pre-routing phase, it will be injected into the normal Freenet network and behave as though the endpoint were the originator of the message.

Protection for data sources is provided by the occasional resetting of the data source field in replies. The fact that a node is listed as the data source for a particular key does not necessarily imply that it actually supplied that data, or was even contacted in the course of the request. It is not possible to tell whether the downstream node provided the file or was merely forwarding a reply sent by someone else. In fact, the very act of successfully requesting a file places it on the downstream node if it was not already there, so a subsequent examination of that node on suspicion reveals nothing about the prior state of affairs, and provides a plausible legal ground that the data was not there until the act of investigation placed it there. Requesting a particular file with a hops-to-live of 1 does not directly reveal whether or not the node was previously storing the file in question, since nodes continue to forward messages having hops-to-live of 1 with finite probability. The success of a large number of requests for related files, however, may provide grounds for suspicion that those files were being stored there previously.

Modification or outright replacement of files by a hostile node is an important threat, and not only because of the corruption of the file itself. Since routing tables are based on replies to requests, a node might attempt to steer traffic towards itself by pretending to have files when it does not and simply returning fictitious data. For data stored under content-hash keys or signature-verifying keys, this is not feasible since inauthentic data can be detected unless a node finds a hash collision or successfully forges a cryptographic signature. Data stored under ordinary descriptive text keys, however, is vulnerable. Some protection is afforded by the expectation that files will be encrypted by the descriptive text, since the node must respond to a hashed key request with data encrypted by the original text key, but a dictionary attack is possible using a table of text keys and their hashes. Even partial dictionaries cause problems since the node can behave normally when an unknown key is requested and forge data when the key is known.

Finally, a number of denial-of-service attacks can be envisioned. The most significant threat is that an attacker will attempt to fill all of the network's storage capacity by inserting a large number of garbage files. An interesting possibility for countering this attack is the Hash Cash scheme[18]. Essentially, the scheme requires the inserter to perform a lengthy computation as "payment" before an insert is accepted, thus slowing down an attack. Another alternative is to divide the datastore into two sections, one for new inserts and one for "established" files (defined as files having received at least a certain number of requests). New inserts can only displace other new inserts, not established files. In this way a flood of garbage inserts might temporarily paralyze insert operations but would not displace existing files. It is difficult for an attacker to artificially legitimize her own (garbage) files by requesting them many times since her requests will be satisfied by the first node to hold the data and not proceed any further. She cannot send requests directly to the other downstream nodes holding her files since their identities are hidden from her. However, adopting this scheme may make it difficult for genuine new inserts to survive long enough to be requested by others and become established.

Attackers may attempt to replace existing files by inserting alternate versions under the same keys. Such an attack is not possible against a content-hash key or signature-verifying key, since

it requires finding a hash collision or successfully forging a cryptographic signature. An attack against a descriptive text key, on the other hand, may result in both versions coexisting in the network. The way in which nodes react to insert collisions (detailed in section 3.2) is intended to make such attacks more difficult. The success of a replacement attack can be measured by the ratio of corrupt versus genuine versions resulting in the system. However, the more corrupt copies the attacker attempts to circulate (by setting a higher hops-to-live on insert), the greater the chance that an insert collision will be encountered, which would cause an increase in the number of genuine copies.

8 Conclusions

The Freenet network provides an effective means of anonymous information storage and retrieval. By using cooperating nodes spread over many computers in conjunction with an efficient routing algorithm, it keeps information anonymous and available while remaining highly scalable. Initial deployment of a test version is underway, and is so far proving successful, with over 15,000 copies downloaded and many interesting files in circulation. Because of the nature of the system, it is impossible to tell exactly how many users there are or how well the insert and request mechanisms are working, but anecdotal evidence is so far positive. We are working on implementing a simulation and visualization suite which will enable more rigorous tests of the protocol and routing algorithm. More realistic simulation is necessary which models the effects of inserts taking place alongside requests, nodes joining and leaving, variation in node capacity, and larger network sizes.

9 Acknowledgements

Portions of this material are based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

- [1] S. Adler, "The Slashdot effect: an analysis of three Internet publications," <http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html> (2000).
- [2] Akamai, <http://www.akamai.com/> (2000).
- [3] American National Standards Institute, Draft: American National Standard X9.30-199X: *Public-Key Cryptography Using Irreversible Algorithms for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA)*. American Bankers Association (1993).
- [4] R.J. Anderson, "The Eternity service," in *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology (PRAGOCRYPT '96)*, Prague, Czech Republic (1996).
- [5] Anonymizer, <http://www.anonymizer.com/> (2000).

- [6] O. Berthold, H. Federrath, and M. Köhntopp, "Project 'Anonymity and unobservability in the Internet'," in *Computers Freedom and Privacy Conference 2000 (CFP 2000) Workshop on Freedom and Privacy by Design* (to appear).
- [7] D.L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM* **24**(2), 84-88 (1981).
- [8] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos, "A prototype implementation of archival intermemory," in *Proceedings of the Fourth ACM Conference on Digital Libraries (DL '99)*, Berkeley, CA, USA. ACM Press: New York (1999).
- [9] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," *Journal of the ACM* **45**(6), 965-982 (1998).
- [10] Church of Spiritual Technology (Scientology) v. Dataweb *et al.*, Cause No. 96/1048, District Court of the Hague, The Netherlands (1999).
- [11] I. Clarke, "A distributed decentralised information storage and retrieval system," unpublished report, Division of Informatics, University of Edinburgh (1999). Available at <http://freenet.sourceforge.net/> (2000).
- [12] R.R. Dingledine, "The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven," M.Eng. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2000). Available at <http://www.freehaven.net/> (2000).
- [13] Distributed.net, <http://www.distributed.net/> (2000).
- [14] D.J. Ellard, J.M. Megquier, and L. Park, "The INDIA protocol," <http://www.eecs.harvard.edu/~ellard/India-WWW/> (2000).
- [15] Gnutella, <http://gnutella.wego.com/> (2000).
- [16] I. Goldberg and D. Wagner, "TAZ servers and the rewebber network: enabling anonymous publishing on the world wide web," <http://www.cs.berkeley.edu/~daw/classes/cs268/taz-www/rewebber.html> (2000).
- [17] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private Internet connections," *Communications of the ACM* **42**(2), 39-41 (1999).
- [18] Hash Cash, <http://www.cypherspace.org/~adam/hashcash/> (2000).
- [19] Mixmaster, <http://www.obscura.com/~loki/remailer/mixmaster-faq.html> (2000).
- [20] Napster, <http://www.napster.com/> (2000).
- [21] M.K. Reiter and A.D. Rubin, "Anonymous web transactions with Crowds," *Communications of the ACM* **42**(2), 32-38 (1999).
- [22] The Rewebber, <http://www.rewebber.de/> (2000).

- [23] M. Richtel and S. Robinson, “Several web sites are attacked on day after assault shut Yahoo,” *The New York Times*, February 9, 2000.
- [24] J. Rosen, “The eroded self,” *The New York Times*, April 30, 2000.
- [25] A.S. Tanenbaum, *Modern Operating Systems*. Prentice-Hall: Upper Saddle River, NJ, USA (1992).
- [26] M. Waldman, A.D. Rubin, and L.F. Cranor, “Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system,” in *Ninth USENIX Security Symposium*, Denver, CO, USA (to appear).
- [27] Zero-Knowledge Systems, <http://www.zks.net/> (2000).