GnuFU

Gnutella For Users

Contents

Editor: ARNE BABENHAUSERHEIDE (http://draketo.de) Beta-Reader:

Website: http://gnufu.net

Note from the editor: Arne Babenhauserheide wrote most of the document. It was opened to the public as a wiki around the 30th of November 2003. Everything which changed since then belongs to the author of that part, and shall be his or her responsibility (as far as that is possible), and his or her copyyright, as long as a summary of the changes and a contact-adress (email, webpage or physical) with nick or name is supplied along.

I thank Janet for very nice proofreading of the english text and enhancements in style and readability.

I thank the whole GDF for two years in which they tolerated me as a non-programmer in their forum because I learned a great deal of the informationn in this document during that time through asking partly quite naive questions (I hope I still managed to contribute there constructively).

Also I thank stief and et voilà from the Gnutellaforums, who are doing their Job on the support side without payment and with by far too few thank you's.

What is Gnutella and how does it work?

Gnutella is an open file sharing Network originally created by Justin Frankel and Tom Pepper of Nullsoft. It was released on the companies webpage without asking its owner AOL and taken down the next day. This did not stop Gnutella, because after a few days the protocol had been reverse engineered (that means people found out how it worked), and compatible open source clones started showing up. This parallel development of different clients by different groups remains the way Gnutella is being developed today.

Being open means, unlike most other networks, everyone can write a client which can access the GNet, if it fulfills the publicly available specifications. The specifications are discussed and created by the »GDF (the Gnutella Development Forum: http://groups.yahoo.com/group/the_gdf), an open Mailinglist for developers with by this date over 1000 members. After that they are documented in the »rfc-gnutella (http://rfc-gnutella.sf.net/). That way all programs share a common base, while the protocol also allows for client specific options. The developers are careful to ensure the greatest possible backwards compatibility.

Despite the name, Gnutella isn't GNU-Software, though some Gnutella clients are GPL-licensed. It is an open network, and the origin of its name may be found more easily by eating too much Nutella, than at GNU (That means: Gnutella is not a project of the FSF or related to GNU software tools).

Back to the main focus of this document: the basic principles of Gnutella, their evolution during the last few years (especially the last year) and future plans for it.

Basics of Gnutella

Network Model: The Original: FoF

You can imagine the original model of the Gnutella network as friends phoning each other to get information. One asks five others, each of whom asks 5 others and so on. After the first step the number of people reached is 5, after the second it is 25, after the 5th 3125, after the 7th 78,125 and after the 14th about 6.1 billion. That would be enough to reach every human being on this planet. The original Gnutella used 7 steps (called HTL: Hops To Live).

A Problem with this model (among others) is that you have to be a part of the clique before you can use it. There have been several ideas to solve this problem. I will show you three of them.

Getting in: The first way: Pong-Caching

Pong Caching means that the node (aka you) asks its friends who their friends are. It means your friends introduce you to their friends, especially friends whom they value highly, and you write all new adresses in your phone-book, so you know whom to phone when your original friends are on holiday (Somehow like being at a continous cocktail party). It is easy and has the advantage of giving you very reliable contacts, but there is no way of getting into

the network without knowing at least one contact who is already in the net. That means you can always get back in, but won't be able to connect if you never did before.

Getting in: The second way: Remember who answers

The second way is really simple. When one of your 5 friends calls back to say Smith (whom you didn't know before) knows something, you note her number. When you call her the next time as one of your five direct contacts, the chance is greater that you will get your information more quickly, because she will likely have friends who have similar interests to you (where else should she have gotten the information?), and those are more likely to have your information than randomly picked persons (at least when you ask about something similar to your last question). The drawback is that those contacts might not be at home often, so it is quite possible that you find a contact with great knowledge, but whom you'll never be able to reach again. Still no way of getting in the first time.
And now we get to one of the recent developements in Gnutella: GWebCaches. I will discuss them in the next part.

Recent Changes in Gnutella

Getting in: The third way: GWebCaches

To stay within the picture, a GWebCache is a contact who puts his phone number into the newspapers and keeps a record of those who call. When you've been away for some time and are no longer certain if your contacts still have the same mobile-phone numbers, you call the publicly known contact. Before giving you numbers, he will ask you: "Do you know other publicly known contacts? If yes, please tell me their numbers." That is done because they can't read all the newspapers, and you do it all the time without working too hard for it. That way, they keep track of each other. Then the contact gives you some numbers to call and notes your number (to give it to someone else) and the adresses of other public contacts he knows (GWebCaches).

This is roughly the way GWebCaches work. As I stated, they are one of the new developements in Gnutella, and thus I will now get to some more of the recent changes within Gnutella and to future plans.

Sidenote: GwebCaches are essential only for the first connection, and only when the local host-cache is empty. They must not be preferred over your local address-book.

Problems of the FoF-model -> Changes

The Friend-of-a-Friend model has certain disadvantages, which have their source in the way searches are performed. If a search brings too many results, the nodes through which you are connected (your nearest 5 friends) can get overloaded, because every answer has to go through them, for they don't give out your "phone number", but their own and hand the answer to you. If you ask for the name of the head of University in the campus, you'll get hundreds of answers in reality, and thousands to millions on the web. Also, if every question is passed to every one in a 75,000 to 600,000 computer-network, and every computer asks only once an hour, each of them has to answer about 130 to 1600 questions per second. And they have to pass them on. While computers are fast, and today's internet connections can handle quite a lot when compared to the

connections a few years ago, this is too much even for them. Just imagine your phone ringing endlessly the whole day for all kind of questions.

To solve this problem, some changes were made to the FoF model.

Network model: Change who calls whom: Ultrapeers and Leafs

-UPs really short

You'll surely have friends who know very many other people, and whom you can ask, and be sure they'll know exactly the person who can give you the answer. These are called Ultrapeers in Gnutella. An Ultrapeer doesn't have to know much herself, she just needs to know who knows it. In Gnutella that means that a good Ultrapeer doesn't need to have many files to benefit the network. If you're afraid to share much, you should become an Ultrapeer in Gnutella

-Ultrapeers and Leafs more detailed

In the Computer World, as in the Real World, there are contacts who can cope with more calls, and those who can't phone often (or can't afford the bills). In the Real World this is likely because they have more free time, whereas, in the Computer World, it is because they have faster connections (Like DSL, Cable, T1, T3 or similar broadband). Upon realizing this, the developers decided to change the topology, that means how the network looks from the outside when you draw it. Now you don't just call any of your friends, but only those of whom you know that they have the time to take your call and to send it on to others. To save you from too many calls, they then ask you which kinds of informations you have or, to express it in a more human way, what your speciality is. In the Computer World that means your computer sends a list of all its files to the Ultrapeer, which is how we call these kinds of contacts. That list doesn't contain the actual names of your files, but data, which allows the UP to check if you might have a file containing a certain keyword. The mechanism for this will be explained in the next part discussing QRP. Whenever a call reaches the Ultrapeer, she checks if you could know an answer and calls you only in that case.

These Ultrapeers have many connections to others, which means they have a big address book. Normally they stay in contact with 16 other Ultrapeers whom they have in their address book and to whom they send questions, and who send them to 16 more, each. Also they have about 16 leafs, who can't or don't want to phone that much, from which they accept calls, and whose files or, for the human world, specialities they know.

This may seem like a foul bargain for the Ultrapeers, who devote far more resources to keeping the network intact than leafs, but in fact it isn't. While the Ultrapeer (UP) uses much of her time for keeping the network running, the leafs specialize on gathering and delivering information. So, when anyone, Ultrapeer or Leaf, wants to know something, he or she simply starts a call and a leaf specialist can explain it to them. That way people specialize to get more for all.

Network model: Intra-ultrapeer QRP

While with Ultrapeers not everyone needs to participate in sending questions to others, and people can specialize in sharing their information instead, the Ultrapeers would still send every question to everyone, without ever taking into account if that UP even has leaves, who have the files. This sounds normal, for how can an Ultrapeer

know which files the other Ultrapeers have? The answer comes, again, from real life. A normal person knows her friends, and knows who of them might know the answer to a specific question, and who most surely will not. In Real Life this is mostly done through friendly chatting.

Now, computers normally don't chat idly, so they don't exchange this information by the way. Thus the Query Routing Protocol was developed. There each Leaf tells its Ultrapeers which files it has, but instead of taking the names, which would consume too much space, each word which is part of the name of a file is saved as numbers (these are computers after all). You can imagine this process like a game of dumping ships (the numbers form the board with two coordinates). An Ultrapeer doesn't send all questions to a leaf, but only those which it might be able to answer (which hit a ship), and so Leafs get far less needless calls.

Now when this takes so much pressure off the Leafs, why not extend it? Exactly that was done. Now all Ultrapeers send their boards to their direct neighbors. They send only those searches, which have one more step to go, to other Ultrapeers on whose board they score a hit. That means, the last two steps of a search will only be taken when there is a chance that they give results. You can see quite simply why this heavily reduces the bandwidth usage by looking at an example: Imagine a tree, a normal tree, not one of those mathematical constructs. If you try to count the leaves, you have almost no chance to ever finish. But if you take the leaves away and count only the branches, you have far less work to do. If you now take away all those tiny branches, you can really begin to count the rest. QRP doesn't take all leaves and all tiny branches away, but it removes those of them who couldn't give you an answer. Since every part through which a question has to travel consumes bandwidth, and there are far more leaves than branches, taking away, in many cases, most of the last two steps (that means many of the leaves and the tiny branches) reduces the number of questions the computers have to send on. The example doesn't work for all of Gnutella, but here it fits nicely. The people of LimeWire talk about 70-80% savings alone through this.

Network model: Change Searching: Dynamic Querying

Now, while the Ultrapeer model and QRP partly solve the problem that you don't have the time to explain something properly to someone else, or to get it explained, because the phone rings endlessly for questions to which you know no answer (or in Tech-Speech: because the network-traffic exceeds your connection-speed), there is still another problem which might normally not even be visible, if you look at it. In the Real World, an Ultrapeer will ask for a specialist who can give you the information until she finds one, and then stop. In the Computer-World, the question is sent on and on, to as many contacts as possible, without looking if there already are answers.

With Dynamic Querying that changes. Now the Ultrapeers ask one other Ultrapeer at a time, and wait a bit, to see if they get answers. When they have enough answers to be satisfied, they stop asking for more. It sounds pretty natural, but was quite a big step for Gnutella because it saves resources which were wasted on very popular questions. I'll take the example of the of the head of university again: now, if you ask for the head of university, your Ultrapeers will first see if they know someone directly who can answer your question. Then they will

simply give you some numbers of people they know who live on the campus. You will still get more than one answer because they will give you more than one number, as they can't be sure that you'll reach every number they gave you. But you won't get thousands of phone-numbers (one from every student on the campus), first because the Ultrapeers would waste their time with that on something which doesn't give you additional benefit, second, because you couldn't ever call all those people, and third, because then you might not reach your Ultrapeers anymore, because they would be too busy getting return calls from others who tell them numbers, and sending your question to other Ultrapeers.
According to the Bearshare Programmers this saves another 60% of bandwidth usage.

Finding sources withough searching aka the Download-Mesh

Now you might say, "but I can't download from those three, because others already do. I want to get all addresses from which I can download," (and you are not alone with this. I feel the same). By looking at the Real World, we can find a solution to this problem, too, without having to waste too many resources on it. There (in the Real World), if you ask a specialist to explain something to you, and that specialist is busy, she will know some other specialists (because they know each other) who might have more time at the moment.

Realizing this concept in Gnutella is not as easy as the Ultrapeer-Leaf Model nor as the Dynamic-Query Model.
But the programmers found a way. As I stated in the Dynamic-Query-Model, you will get more than one number at which you can ask. Now, when you call someone who should know the answer, you also give her the other numbers you know about. That way, the specialists will get to know each other (the same way, as the GWebCaches, which I mentioned before, learn of others of their kind). As everyone who asks also brings her own set of numbers, the specialists know more and more additional addresses, and when you ask them to explain, and they don't have time at the moment, they give them to you (they do it even if they have time, just in case they could be interrupted, and because in Gnutella you can download from more than one source at once, just like you can in the overnet-network (which does this to the extreme, but is only really efficient for big files)). Additionally, the specialists also add you to their list of alternate-contacts, as soon as you know enough to teach others.

This is why often many people download files from you which you just downloaded yourself.

Better Downloading Part1: Swarming and Partial File Sharing

Swarming is quickly explained (but hard to do in the friend-of-a-friend model, so I drop it for this part only). It works by simply getting one file from more than one person at once. The file is separated into several parts, as if you'd want to get a book from some friends and every one of them copied only a few pages of it. When you ask every one of them to copy a different part of the book, you'll get the complete book, and every friend of yours has only very little work to do (and if one doesn't have the time to do it, another one can).

Swarming works best with the Download Mesh and Partial File Sharing (PFS), which allows people to share files which they are downloading at the moment, because they

can share those parts which they already have, while they still download from others. You can copy those pages which you have without having to have the whole book, since your pages are all numbered and your friends can also ask you for certain page numbers.
The name has no further meaning, but nicely conjurs the images of antlike fileparts swarming to your Computer.

Better Downloading Part2: Downloading through Firewalls

Imagine there were people who couldn't be called, but could only call others (maybe because they only use public phones, or their number isn't displayed on your phone, and they don't like to give it out because they don't like being called by telemarketers or by people terrorizing them over the phone). In Gnutella these are computers who are behind a firewall. They can call others and get information from them, but no one can call them.

The solution is to have the firewalled people call their Ultrapeers regularly, and when someone wants to call them, she simply calls the Ultrapeer who then holds two phones together, one to which the firewalled person (the one who can't be called) raised a call, and the one you called. That way you can talk to the firewalled person, but it takes two simultaneously running calls, which means, that it needs twice the bandwidth in the Computer-World. Firewalled persons always keep their connection to the Ultrapeers, who simply relay the information or data.

There are plans to save the Ultrapeers from this additional bandwith usage by letting other people do the phone connecting. Then, when someone wanted to get information from a firewalled specialist, the Ultrapeer would tell the firewalled person and the asker to call a third person. That person would then hold the two phones together. In Gnutella most People have three to five phones, so this wouldn't be such a great problem. These phone-connectors will most likely be called routing-peers.

File-Magnets

File-Magnets stray from the Friend of a Friend model. They are links on webpages, which you can simply click, and which will tell your file-sharing program to search Gnutella (in fact also other networks) for a specific file, and to download exactly this.

You can imagine it like an article in a newspaper which tells you information which gives your Ultrapeers the exact information that the specialist, from whom you want to learn, has to know. In the Real World you would most likely find one specialist and those who learned from her.

With a magnet-link you can avoid getting bad files because they use a hash-string, which is something like a summary of the information the specialist would give you. If she begins to tell you crap, you will see at once that it doesn't fit the summary. In Gnutella, the program asks for files to which the people who have them have assigned the same summary, aka Hash-string. After downloading, the program does its own summary and checks if they really match. If not, it tells you that the file is corrupt. The summaries from same files are always exactly the same because they are done via specific mathematic methods which always get to the same result when given the same data (aka information) (Sha1-Hash at the moment).

Different from Magnet-Links, KaZaA-Links and eDonkey-Links are not secure, because they use methods which can be betrayed with false files (for example a KaZaA-Link asks for a kind of summary which only checks the introduction and the first part of the information, but all the rest is ignored to make the summary quicker to create. Naturally it is very easy to give you false information, because specialists only have to tell the truth at the beginning, then they can lie or fantasize as much as they want). Further information on Magnet-Links can be found here: »Magnet-Uri (http://magnet-uri.sourceforge.net/) and on »http://www.MagnetLink.org.

There is now a new version of magnet-links: KaZaA magnets. Sadly those might also not be secure, for they use the KaZaA hashing system (the incomplete summary) with some changes (they now add another smaller summary, which might tell you about the missing parts, but they didn't publish, how they create it). If KaZaA-Magnets provide information about a search term, they might work with Gnutella, but they won't ensure that you get what they offer to you. If you find the word "kzhash" in the link, it might not be secure (aside from having a somewhat misplaced name).
You'll find some magnet-links on the pages listed in the MagnetLists-page (http://magnetlists.gnufu.net)

Future Plans

•A Community-Feature.
•Encryption - Making it impossible for the provider to see what you are downloading. Bearshare already does this for communicating with other Bearshare-clients.
•Caching of popular content.
•Magma-Lists - Multiple-file-magnets, like playlists and with additional information
•Privacy (See AnoGnut: http://anognut.gnufu.net).
•Routing Peers and better Firewall support.
•What's New? - Finding New Files in Gnutella (implemented by LimeWire).

How to get in?

To become part of the Gnutella Network, you can use one of the clients listed on http://www.gnutella.com/connect/
(should that be down, just use the list on: »dmoz (http://dmoz.org/Computers/Software/Internet/Clients/File_Sharing/Gnutella/), or better still, learn to program and help furthering the development of some of the open-source clients.

On the Website of GnuFU you'll also find Links to several Open Source and Closed Source Gnutella Clients: http://gnufu.net

As the page hosting GnuFU was down two times during the last week, I decided to no longer depend completely on any provider to host this page. Should GnuFU go down due to any reason, a freenet-mirror of the PWP-code can be found using the following key:
SSK@1~6U-1UApvA5hld50tMsau3O5tEPAgM,kr~pLjSLxfECXC2Mvt3RKw/gnufu/3//index.html
Most times the link "http://free.gnufu.net" should get you there.
To access this mirror you need to obtain freenet from »http://freenetproject.org.

GnuFU

Document created by Arne Babenhauserheide
using Ragtime (http://www.ragtime.de)
Last changed: 22. Jun 2004