Bootstrapping Gnutella Using UDP Host Caches

Sjoerd Langkemper
stlangke@cs.vu.nl

May 19, 2005

Abstract

In peer-to-peer networks, like Gnutella, an overlay network is used where nodes connect to each other. Not all nodes in a network support the overlay network, so a joining node has to have a way to find nodes already in the network. Finding good nodes and connecting to them is called bootstrapping and is a problem in modern peer-to-peer networks.

Thus far, advertising nodes which are already connected to the Gnutella network was done out of band, for example using HTTP. These methods have major problems with scalability and reliability. This paper proposes a solution called UDP host caches, which makes use of the Gnutella protocol itself.

1 Introduction

Since its creation in 2000, Gnutella has experienced an exponential growth. Because of this, scalability has always been a large issue. One of the creators of Gnutella complained when 5000 hosts were trying to connect at the same time, while today the Gnutella network is used by millions of users.

A particular limit on scalability is the bootstrapping of clients. Bootstrapping is the process of connecting to the network. Since the topology of the Gnutella network changes constantly, finding nodes that are already connected to the network can be a problem. When a client has not connected for a long time, or when it connects for the first time, it needs a way to get addresses of nodes that are already connected.

In 2002, GWebCaches were set up to inform the clients were other clients could be found. These caches could not keep up with the growth of Gnutella and relatively few working GWebCaches can be found today. [Kar03] have shown that there are not only too few GWebCaches, they are also bad quality.

This paper shows how to solve the bootstrapping problem in a scalable way, using UDP Host Caches. This will decrease the time it takes to connect to the Gnutella network.

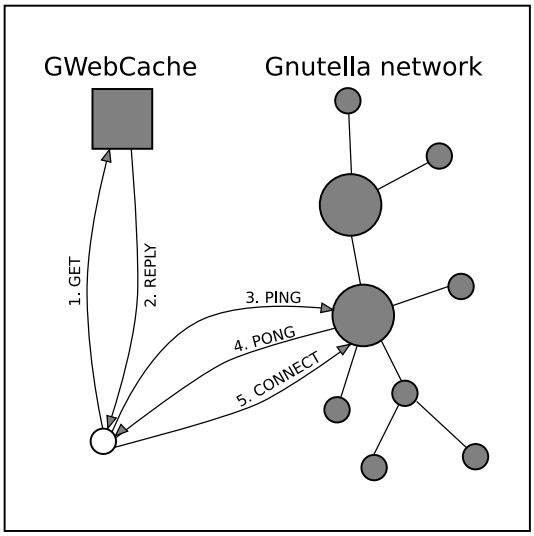

Figure 1: Typical use of a GWebCache.

2 Background

One of the first widely used peer-to-peer networks, Napster, made use of servers to advertise nodes. In most peer-to-peer networks, installing one server which only responsibility it is to advertise nodes would be trivial, but would have great disadvantages. First, this server will become a bottleneck for the entire network. There are many developers of Gnutella software and if one application misbehaves this could take up all the bandwidth and processing power on the server.

Second, one server introduces a single point of failure. Some failures can be solved with a backup server. In the Napster case, however, the server administrators received a court order to shut down all servers, including backup servers.

Gnutella currently makes use of GWebCaches. These are web pages with lists of hosts already connected to the network. According to [Atk05], there are 20 GWebCaches taking more than 700.000 requests per hour[1], which is almost 200 requests per second. Because these pages are dynamic and each host must be checked before it is inserted into the host list, one needs a good webserver and much bandwidth. Most Gnutella users do not have the knowledge or the bandwidth to run a Gnutella cache.

[Web03] suggests using IRC, web search or Usenet (newsgroups) to advertise hosts. Web search is too slow: Gnutella clients run for hours and indexing by a search engine may take weeks. Usenet is not capable of sustaining the load. It is build for 200 messages are day, not 200 messages a second. Providers of Usenet servers would not like this extra load on their servers. IRC could work, but the IRC system is also not made for the job and Gnutella users would be imposing a great load on the IRC system without the providers consent.

[1]The number of requests per hour vary throughout the day, so this is only an indication.

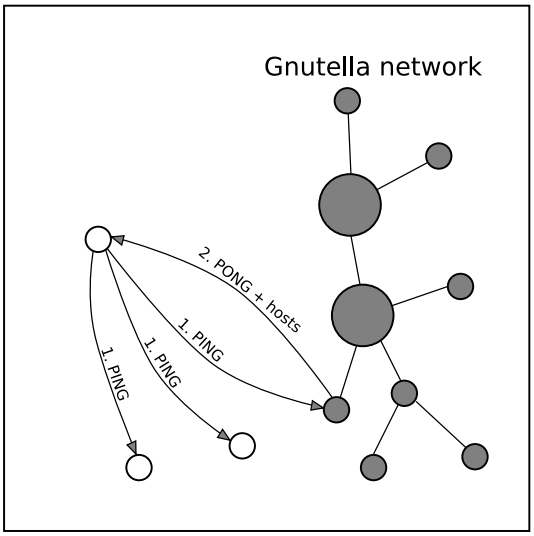

Figure 2: Workings of UDP host caches when the connecting client has a list of nodes which are possibly connected to the network.

Another method, much used in the past and also put forward by [Web03] would be to randomly scan hosts to see if they support Gnutella. This may be considered as a port scan, which is illegal in some countries. Furthermore, it becomes very hard when IPv6 is introduced, because of the much larger address space.

3 Presented solution

The presented solution is called UDP host caches. Node information is integrated in the UDP PONG packets that are normally transfered within a Gnutella network.

Imagine node A is trying to connect to the Gnutella network and it has a list of hosts which may be already connected, as in figure 2. In this case, node A sends PING packets to verify that the nodes are online. PING packets are small UDP packets which are designed for this purpose. All nodes that are up respond to node A with a PONG packet, indicating that they are ready to accept a connection.

With UDP host caches, a list of hosts is embedded in the PONG message. Thus when node A finds a node which is accepting connections, it gets a list of other nodes to try. If node A can find one node which responds to the PING message with a list of hosts, it can connect to the network easily.

The problem remains of finding the first node. When a Gnutella application exits, it stores the nodes it was connected to in a file, so that it can try to connect to these hosts the next time. If none of these nodes are up, or if the client is connecting the first time, the node can send a PING to a well-known UDP cache. This UDP cache can be set up by the provider of the application and be put in the configuration of that application.

UDP messages are lightweight, so the sending of these messages takes not much bandwidth. A request to a GWebCache takes at least 6 packets and approximately 704 bytes[2], where a request to a UDP host cache takes 2 packets and approximately 233 bytes[3].

Since PINGs and PONGs are already sent by all Gnutella applications, it is trivial to support UDP host caches. Every Gnutella client can thus become a UDP host cache. When a node meets certain conditions, such as a long uptime, it can advertise itself to be used as a UDP host cache. Where a GWebCache needs to be set up by a user, a UDP host cache can be enabled without the user even noticing it. This increases the number of caches and decreases the load on each of them.

4 Evaluation

GWebCaches could work well for bootstrapping the Gnutella network. The problem is that certain applications misuse the GWebCaches. A client should not have to connect to a GWebCache more than once a day. The rest of the time it should use the list of hosts which it used the previous time.

According to [Lim05], there are approximately 1.4 million nodes connected at any time. If they each would connect once a day, that would result in 58.333 requests per hour, instead of the 700.000 reported by [Atk05] . Some clients query GWebCaches much too often, unnecessary imposing load on the servers. Furthermore, if the quality of the GWebCaches improves, clients would need to connect only once to get enough nodes. Twenty GWebCaches would be enough if the load was divided by ten.

UDP host caches are probably also going to be misused by poorly written applications. However, they use UDP and transfer IP addresses in binary instead of ASCII, which uses less bandwidth. UDP host caches are therefore much more resistant to high loads.

5 Conclusion

UDP host caches are much more lightweight than previous solutions. They are also integrated in the Gnutella protocol. This ensures that they are capable of handling the load of the current Gnutella network and also are scalable enough to work well when the Gnutella network increases in size. Furthermore, since UDP host caches can be set up automatically, the number of caches increases as the size of the Gnutella network increases.

Because UDP host caches are more lightweight and there can be many of them, the time it takes to connect to the Gnutella network will decrease.

[2]Each TCP/IP packet accounts for 44 bytes, GET request for 100 bytes, reply for 340 bytes (average of 517674 replies).

[3]Each UDP/IP packet accounts for 32 bytes, PING for 30 bytes, reply for 139 bytes (average of 51 PONGs).

References

[Sla00] Slashdot.org - *Open Source Napster: Gnutella* - http://slashdot.org/article.pl?sid=00/03/14/0949234

[Ber04] Sam Berlin in the Gnutella Developer Forum - http://groups.yahoo.com/group/the_gdf/message/20840

[Atk05] Gnutella Webcache Scan Report - http://gcachescan.jonatkins.com/

[Lim05] Today's Host Count - http://www.limewire.com/english/content/netsize.shtml

[And01] *Analysis of the Traffic on the Gnutella Network* - Kelsey Anderson

[Kar03] *Bootstrapping in Gnutella: A Measurement Study* - Pradnya Karbhari et. al.

[Tay03] *Lecture 3: Gnutella* - Ian Taylor

[Web03] *Advertising Peer-to-Peer Networks over the Internet* - Matthieu Weber et. al.