Decentralized Peer-to-Peer Network Architecture: Gnutella and Freenet

AUTHOR:

Jem E. Berkes

umberkes@cc.umanitoba.ca

University of Manitoba

Winnipeg, Manitoba

Canada

April 9, 2003

Introduction

Although traditional network file systems like NFS provide a reliable way for users on a LAN to pool and share data, Internet-wide file sharing is still in its infancy. Software developers and researchers are struggling to find new ways to reliably, efficiently and securely share data across wide area networks that are plagued by high latency, bottlenecks, and unreliable or malicious nodes.

This paper will focus on decentralized file sharing networks that allow free Internet-wide participation with generic content. A decentralized network has no central authority, which means that it can operate with freely running nodes alone (peer-to-peer, or P2P). Much of the current research into file sharing focuses on such systems, after the repeated failure of commercially-oriented networks such as *Napster*™ and *Morpheus*™ demonstrated that centralized and purely multimedia-based systems were unsuitable for long-term use by the general Internet public.

Building a useful decentralized file sharing network is no small feat, but an effective system will be a remarkable accomplishment for the modern Internet. A robust, massive content distribution network will have a multitude of uses. The huge amount of accessible data (already into hundreds of terabytes on existing networks) and enormous transfer capacity at little or no cost to individual participants demonstrates the value of such a system, which may soon become a core Internet technology akin to the World Wide Web. Such large, anonymous networks seem quite natural for the Internet, as they demonstrate the epitome of pooling resources for the mutual benefit of all users.

The first major project to delve into decentralized file sharing was *Gnutella*, developed by the Nullsoft team (also developers of the *Winamp* media player). After being posted in 2000, the software was quickly removed from the web site by Nullsoft's owners, America Online Inc., and the plans to release the specification of the protocol were abandoned. Nevertheless, other developers were able to reverse engineer the protocol and publicly released the specification. The publication of a well defined protocol specification (currently *Gnutella v0.6*) proved to be extremely useful, as different developers were able to contribute their own Gnutella-compliant software that could inter-operate.

Another prominent decentralized file sharing system is *Freenet*, an open source implementation of a system described by Ian Clarke in 1999. *Freenet* differs from *Gnutella* in that its primary purpose is to create an uncensorable and secure global information storage system. The *Freenet* architecture is designed with special consideration for anonymity and fault-tolerance.

This paper will investigate the architecture and protocol details of the *Gnutella* and *Freenet* systems. Real-world considerations and practical extensions such as firewall compatibility are also examined.

Gnutella 0.6 architecture

Gnutella is a decentralized peer-to-peer system, consisting of hosts connected to one another over TCP/IP (default TCP port: 6346) and running software that implements the Gnutella protocol. This connection of individual hosts (or nodes) forms a network of computers exchanging Gnutella traffic, consisting of: queries, replies to queries, and other control messages used to discover nodes. This network allows the participating hosts to share arbitrary resources. For example, resources may be mappings to other resources, meta-information, or other kinds of pointers. Current implementations focus almost exclusively on data files, meaning any given host can offer local files for others to download, and can also download files that others are sharing on their computers. Each node is controlled by a user running the application software, who participates in the network by:

- Explicitly specifying a list of local files to share across the network
- Searching for files existing somewhere on the network
- Downloading files from other nodes

A node that wishes to participate in the network must join the Gnutella network by connecting to any existing node. Currently, mechanisms known as “host caches” allow new nodes to easily participate in the network by connecting to a random node's IP address provided via DNS or script running on a web site. The new node opens a TCP/IP connection to the existing node and performs a handshake (note that this is at the Application layer; this is all over TCP). Part of the handshake includes an exchange of client and server capabilities, since different clients may choose to implement different features. The handshake is modeled after HTTP and familiar features such as the User-Agent header and response codes exist. If the server decides to refuse a new client (e.g., with the 503 Busy response), it is encouraged to provide an X-Try: header that recommends other IP/port addresses to try connecting to.

A node typically connects to multiple existing Gnutella nodes to be able to reach more total Gnutella nodes. Once a node has connected to the network, it communicates with its neighboring nodes by sending and receiving Gnutella protocol messages and accepts incoming connections from new nodes that wish to join the network (it listens for connections on the appropriate TCP port). The main network messages are summarized in the table below.

Type	*Description*
Ping	A request for information about another node(s)
Pong	A reply carrying information about a node (e.g., number of files shared)
Push	A mechanism that allows a firewalled node to share data
Query	A request for a resource (e.g., searching for a file)
Query Hit	A response identifying an available resource (e.g., a matching file)

Table 1: Gnutella protocol messages ('Bye' excluded)

Gnutella is a broadcast-type network, in which Pings and Queries are duplicated and relayed to multiple other nodes. To reduce network resource consumption, nodes cache Pongs and supply them as responses to Pings when they can. Pongs and Query Hits are routed by each node back along the path needed to reach the destination. In this respect, requests are very inefficient (due to the broadcast or flood nature) but replies are directed back to unique nodes rather efficiently.

The Ultrapeer is an important concept that was not specified in the original Gnutella protocol, but which has now become a prominent feature of the Gnutella network. The Ultrapeer scheme improves network efficiency and scalability by categorizing nodes into “regular clients” and “super nodes”. A super node is a reliably connected host with plenty of network bandwidth that can act as a proxy for a large number of connecting clients. The super node removes the burden of extensive network message routing from the client, which may be a low bandwidth modem user. In such a case, the modem user (a leaf node) uses the well connected super node as an entry point into the network. When used in such a hierarchal fashion, the Ultrapeer concept lets the Gnutella network scale quite well since it vastly reduces the number of nodes actually involved in message routing. With this scheme, the Gnutella network mimics the Internet itself: low bandwidth nodes are connected to larger routers (the super nodes) that transmit the majority of the data over high bandwidth “backbones”.

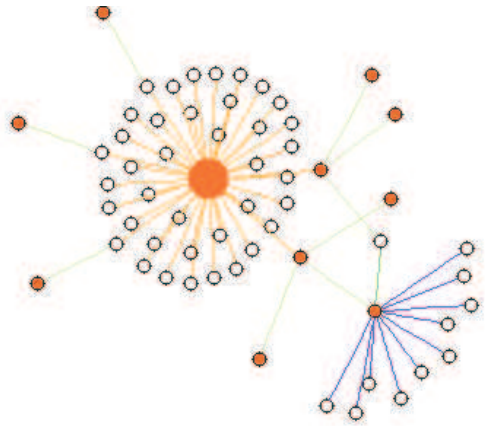

Figure 1: Actual Gnutella node map. Super nodes (solid) act as proxy for many leaf nodes (hollow, some not shown)

Freenet architecture

Freenet has a similar network topology to Gnutella, except that its primary purpose is to create a secure global information storage system. Key here is the term "storage system". Freenet nodes not only route messages around the network, but also store various files that don't belong to themselves. In this way, the Freenet creates a very large, distributed file storage system that has plenty of redundancy. The network is quite fault-tolerant, since many nodes store the same data.

The high level architecture of the Freenet resembles that of Gnutella. Individual nodes are connected together by TCP/IP connections, and a new node connecting to the network needs to know about existing node(s) in order to successfully connect. Because there is no central authority, clients must find the first node on their own using a host cache or knowledge from a friend.

The user who controls the Freenet software can interact with the network by:

- Explicitly specifying disk storage space that will be used to store some *unknown* network data
- Sending an "insert" message to add a new file to the network (starts distributed storage)
- Sending a "request" message with a key to retrieve a specific file from the network

This is quite different from Gnutella, since a user can not actually access any of the encrypted data stored locally. Nor can they request content from specific nodes; all they can do is send a request into the network and wait for some node to return the desired data. Furthermore, users can not search the network for arbitrary files. They must already know the key describing the desired file (perhaps supplied by a friend).

The Freenet architecture may seem quite strange, but its purpose is to create a secure network. Each node only knows about its immediate neighbors ("worms eye view"), so it is impossible for a node to determine where on the greater network the desired data is actually stored. In this respect, the network is very resistant to tampering and censorship.

Freenet extensively uses one-way secure hashes such as SHA1, meaning that a unique identifying hash (fixed size) can be generated for any file, but given only the hash it is computationally infeasible to determine what data it represents. Nodes route and store these hashes, so they are not aware of the actual data contents the hashes represent. Furthermore, data stored on each node is encrypted with a key that is not available to the node. Thus, each node is completely ignorant of the nature of the content they are storing for the network. This is an essential feature of the secure system.

Searching Gnutella

Users on the Gnutella network can search for content by sending arbitrary queries into the system. The node that wishes to begin the search sends a Query message to all nodes connected directly to it. Each of these nodes will in turn replicate and send the Query message to its own neighbors, except the node that originated the query. This process continues, flooding (or broadcasting) the query to every node within some radius.

Because queries occupy so much of the Gnutella network bandwidth, the specification restricts the maximum size of a Query message to 256 bytes. Additional measures are in place to make sure that the queries do not uselessly and endlessly traverse the network. Gnutella messages have associated TTL (Time To Live) values, which are decremented at each hop (much like in IP networks). Nodes silently drop the queries once the TTL field reaches 0. Nodes are also responsible for dropping queries that they see twice.

Queries include a "minimum speed" field, which directs nodes to reply to the query only if they can satisfy the minimum transfer speed for file transfers. The Query also consists of a "search criteria" field and extensions block, allowing clients to search for files based on a number of criteria, sometimes client-specific. The generic query is based on a series of keywords, meant to locate files that match all keywords (e.g., "slackware", "iso" matches "slackware-9.0.iso"). Additionally, a special type of query exists which requests a list of all files served by a node.

A node replies with a Query Hit message when it has content that can satisfy the request. Most importantly, the Query Hit contains an IP address and port number where this specific node can be reached via TCP connection for the actual file transfer. These Query Hit messages are only sent along the same path that carried the incoming Query message.

Searching Freenet

As mentioned earlier, Freenet differs from Gnutella in that a node can not search for arbitrary files. A node that wishes to "find" content must already know what file it wants (via a SHA1 generated key that uniquely identifies that file). To retrieve a file, the user sends a request message into the Freenet that contains the globally unique identifier (GUID) key and waits for the data to arrive.

Freenet avoids the wasteful broadcast-style search method that Gnutella uses by relying instead on a "steepest-ascent hill-climbing" search. Each node forwards queries to the neighboring node it thinks is closest to the target. This type of decision making is achieved via a routing table maintained

(privately) by each node. This table lists the addresses of other nodes and the GUID keys they may be storing. When a node sees a reply to a GUID key, it records the nodal direction the reply came from. Data storage locations are not absolute; the network search is interested in the *direction* to go in order to find data. Nodes that pass replies to requests may also store the data locally, themselves becoming future sources for that data.

"Request failed" messages may be passed back to the sender if dead-ends are encountered. Given such signals, nodes that are routing requests will try their next best guesses for directions in which to search. Throughout this process, each node updates its routing table to record the next node to go to when trying to find a certain key. In this respect the Freenet is quite intelligent compared to Gnutella, since a request homes in closer with each hop. And since nodes remember past search results, subsequent queries for similar keys will be directed along paths that are more likely to quickly find the key. The Freenet "learns" how to efficiently locate files through this training process.

Gnutella file transfer

Once a node receives a Query Hit message that satisfies its request, it knows where to find the file it wants. An important point about Gnutella is that files are downloaded out-of-network: the two hosts involved in the transfer connect over TCP/IP and transfer the data directly, instead of wasting the Gnutella network capacity with bytes from files.

The file download protocol is HTTP/1.0 or HTTP/1.1, the standard protocol for downloading files from web servers. The details of HTTP are beyond the scope of this paper, and can be found in RFCs 1945 and 2616 for versions 1.0 and 1.1, respectively. The node wanting to download a file makes a TCP/IP connection to the serving host at the IP address and port number specified, then makes a standard HTTP request. This scheme is modified slightly if the serving host is firewalled.

Freenet file transfer

While Gnutella saves network bandwidth by transferring files out-of-network, the Freenet does transfer the files entirely within the network. This is primarily for security reasons, since the Freenet aims to protect the actual sources of the data from being discovered. Each node only knows the Freenet as far as its neighboring nodes. So the actual files could be coming directly from one of the node's neighbors, or they could be coming from X hops away from its neighbors. There is no way to know for certain.

Large files on the Freenet can be split into multiple pieces. Each file piece has an associated content-hash key (CHK) that can be used to verify integrity of the file data. After transferring a file, or a piece of the file, the receiving node hashes the contents of the file with SHA1 and ensures that the result matches the original CHK. The original CHK is known for certain because this is the actual "key" used to request the file in the first place. Thus after performing a download and verifying that the file hashes match, a user can be absolutely certain that (according to all modern cryptographic knowledge) the received file is an unaltered copy of the one requested.

Considerations and practical extensions

This section addresses real-world considerations and practical extensions that are used by just about every modern peer-to-peer file sharing program. One of the remaining challenges for P2P software is ensuring that different clients use the extensions in a compatible way.

Firewall-friendliness is a difficult problem to solve, because Internet users who are behind firewalls can not accept incoming TCP connections from the outside world. They can certainly join P2P networks by connecting to other hosts, and they can exchange network messages over the established TCP connections. However, they can not themselves allow other Internet hosts to connect to them in order to join the network, which challenges the growth of P2P networks in the long term.

File transfers can be a problem if two hosts have to contact each other directly. If the host serving the file is behind a firewall, other nodes can not connect to it in order to download the file. A partial solution offered by Gnutella is the **Push** message. The node that wants to download data from a firewalled host sends a push message through the Gnutella network to that host. The firewalled host then receives the message, and originates a TCP/IP connection back to the requesting node. Provided the node wanting the file is itself not behind a firewall, the file transfer can then proceed.

File Hashing is a solution that can ensure perfect file transfers. Known hashes (e.g., MD5 or SHA1) can be advertised for files before transfer. The received file is hashed after transfer and compared to the known hash; if the values agree, the file is virtually guaranteed to be error-free.

Transfer recovery and **multi-source downloads** go hand in hand to offer more flexible downloads. If the transfer protocol includes a mechanism to reference offsets within files (as HTTP/1.1 does), then a client can request a "missing" part of a file to resume a download. A client can also simultaneously request distinct parts of a file from several serving hosts, resulting in a particularly high data transfer rate. This is sometimes referred to as "swarming".

References

Cable News Network. (2000). Open source Napster-like product disappears after release. Available on the World Wide Web at http://www.cnn.com/2000/TECH/ptech/03/15/gnutella/

Clarke, I., Hong, T. W., and Sandberg, O. (2002). Protecting Free Expression Online with Freenet. (IEEE) Computer, Jan/Feb 2002: pp. 40-49.

Freenet Project Inc. (2003). The Free Network Project. Available on the World Wide Web at http://freenetproject.org/

Gnutella Developer Forum. (2003). The Gnutella Developer Forum (GDF). Available on the World Wide Web at http://groups.yahoo.com/group/the_gdf/

Klingberg, T. and Manfredi, R. (2002). Gnutella 0.6. Retrieved from the World Wide Web at http://groups.yahoo.com/group/the_gdf/files/Development/GnutellaProtocol-v0.6-200206draft.txt