Design Document – Gnutella Implementation ChakNode v0.1 Sovann Chak

ChakNode Class

The ChakNode is my personal implementation of protocol for the Gnutella distributed search. A ChakNode can be described (in terms of the Gnutella Protocol) as a servent or servant. Each ChakNode acts as a client and server – as is the nature of a peer-to-peer service, and also provides a client-interface to issue queries and view/receive results.

To properly simulate a network, the ChakNode's were designed with the idea of all being hosted from the same network (localhost or 127.0.0.1) and the main distinction between nodes would be a unique port number. Allowing a network of ChakNodes to be simulated from a single machine.

When calling the default constructor on a ChakNode, various actions occur:

- Random selection of an open port between 1024-10000
- A unique Id is generated for the node
- A temporary directory and temporary text files with randomly selected quotes are generated

Once the user joins the network, two threads begin spinning one for periodic pinging and one for listening. The periodic pinging will begin to look within the initial radius. The radius is of size 100. One way of looking at the network would be on an x and y axis where x and y are equal and represented by the port number. The ChakNode looks around its radius to search for nodes. If no node is found, then the radius will continue to grow until the ChakNode's capacity has been reached. For example, if your ChakNode has a port of 1124, the discovery thread will look for potential neighbors by pinging ports [1024, 1224]. The user also has the choice to ping whatever node the wish.

ChakNodeNetwork Class

The ChakNodeNetwork file was designed with the idea of starting a large network of nodes very easily. There are various options provided to a user and provides a nice over view of the network. Since these nodes are being created on localhost, it can interact with any other node on the network including nodes instantiated directly from the ChakNode class.

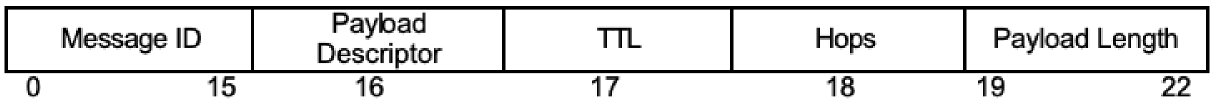
Protocol Specification

The ChakNode protocol is very similar to the Gnutella v0.4 protocols with a few exceptions in regards to the queries and query hits. The following table is a high-level view of the ChakNode protocols

Descriptor	Description
ping	Used to actively discover hosts on the network. A ChakNode receiving a Ping descriptor is expected to respond with one or more Pong descriptors. This is also to perform a well-ness check on existing neighbors, and no response is met with a removal from the neighbor list.
pong	The response to a Ping. Includes the address of a connected Gnutella servant and information regarding the amount of data it is making available to the network.
query	Used to search for a specified string and if a match is found, it will let the requesting node know by sending a queryHit.
queryHit	Used to forward the byte data of the file and a request id. The request id provides a level of anonymity for the downloader of the file as well as the node which had stored the file.

Once a node obtains some port on the network, the first thing that is sent is a descriptor header. The descriptor header contains various pieces of information including which type of descriptor is being sent, and the potential size of a payload:

Descriptor Header



Properties	Description
Descriptor ID	A 16-byte string uniquely identifying the descriptor on the network
Payload Descriptor	0x00 = Ping 0x01 = Pong 0x70 = Query 0x71 = QueryHit

TTL	The number of times the descriptor will be forwarded by Gnutella ChakNode before it is removed from the network. Each ChakNode will decrement the TTL before passing it on to another ChakNode. When the TTL reaches 0, the descriptor will no longer be forwarded.
Hops	The number of times the descriptor has been forwarded.
Payload Length	The length of the descriptor immediately following this header.

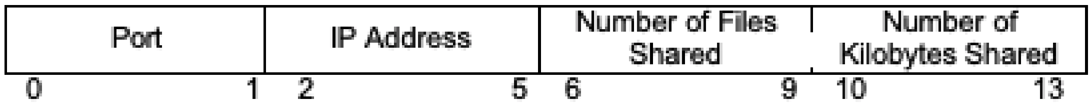
A descriptor header will always be the first aspect of any packet, so the ChakNode can properly understand what steps are next (i.e. ping, pong, …).

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 also 0x00.
Once a ping is realized, a ChakNode will look for a message GNUTELLA CONNECT\n\n, if the requesting node is satisfactory, then

A ChakNode uses Ping descriptors to actively probe the network for other ChakNodes. A ChakNode receiving a Ping descriptor will look for a message GNUTELLA CONNECT\n\n and then may elect to respond with a Pong descriptor, which contains the address of an active ChakNode and the amount of data it's sharing on the network.

Pong (0x01)

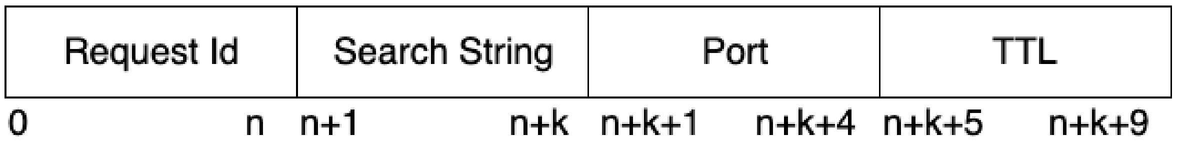


Properties	Description
Port	The port number on which the responding host can accept incoming connections.
IP Address	The IP address of the responding host (in our the case of the ChakNode it is always localhost)
Number of Files Shared	The number of files that this ChakNode is sharing on the network.

Number of bytes Shared	The number of bytes of data that the ChakNode with the given IP address and port is sharing on the network

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. It is also valid for the ChakNode to not elect to respond with a pong descriptor.

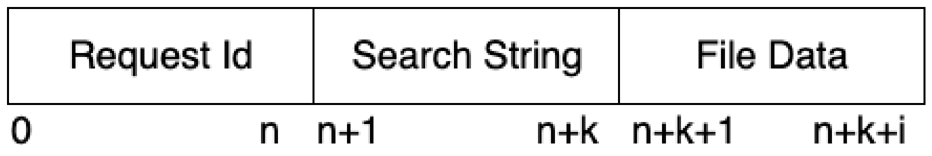
Query (0x70)



Properties	Description
Request Id	A unique id which is generated once a search initially begins. If some ChakNode has already received this request id, it will drop the query. Otherwise, it will search locally for the file and if not found, it will forward the query to its neighbors.
Search String	The exact name of the file that some ChakNode is looking for.
Port	The port number on which the responding host can accept incoming connections.
TTL	The number of times the descriptor will be forwarded by Gnutella ChakNode before it is removed from the network. Each ChakNode will decrement the TTL before passing it on to another ChakNode. When the TTL reaches 0, the descriptor will no longer be forwarded.

A query descriptor is sent when a node cannot find a file locally. This descriptor will be forwarded to each of its neighbors in attempt to find the file which matches the Search String. Each time a query is sent, the TTL is decremented. If this hit zero, the query will not be forwarded any further. If a file is found, a QueryHit descriptor will be forwarded to the port which requested the query.

QueryHit (0x71)



Properties	Description
Request Id	A unique id which is generated once a search initially begins. If some ChakNode has already received this request id, it will drop the query. Otherwise, it will search locally for the file and if not found, it will forward the query to its neighbors.
Search String	The exact name of the file that some ChakNode is looking for.
File Data	The byte data of the file to be downloaded on the requested node.

A QueryHit descriptor is only sent when a file has been found locally, and until the requesting node receives the QueryHit, the descriptor will be continuously be forwarded to the port which forwarded the request. Once the ChakNode with a port matches the port that forwarded the request – the ChakNode will then download the byte data and create a copy of the requested file.

Descriptor Routing

The proper routing is discussed in the Gnutella v0.4 Protocol Specification and the ChakNode Protocol follows an almost identical Descriptor Routing rule set with some minor changes. The rules reiterated and slightly edited are as follows:

1. Pong descriptors may only be sent along the same path that carried the incoming Ping descriptor. This ensures that only those ChakNodes that routed the Ping descriptor will see the Pong descriptor in response.

2. QueryHit descriptors may only be sent along the same path that carried the incoming Query descriptor. This ensures that only those ChakNodes that routed the Query descriptor will see the QueryHit descriptor in response.

3. A ChakNode will forward incoming Ping and Query descriptors to all of its directly connected ChakNodes, except the one that delivered the incoming Ping or Query.

4. A ChakNode will decrement a descriptor header’s TTL field, and increment its Hops field, before it forwards the descriptor to any directly connected ChakNode. If, after decrementing the header’s TTL field, the TTL field is found to be zero, the descriptor is not forwarded along any connection.

5. A ChakNode receiving a descriptor with the same request id and request id as one it has received before, should attempt to avoid forwarding the query descriptor to any connected ChakNode. Its intended recipients have already received such a descriptor, and sending it again merely wastes network bandwidth.