Gnutella

This document describes the implementation of a simulation of a Gnutella network.

For more information about the usage, see the readme file.

Nodes

Each node or server of the network is described in a file called node-[1-8].yaml. These nodes have their own warehouse where their files are stored. On startup, a node scans his folder to build a list of available files. When receiving a query request from a client, it saves it and check if he has the file in his warehouse. In the same time, the request is forwarded to all its neighbors. When a node receives a query, its uuid is stored so the request is processed only one time. If a coressponfing file is found in its warehouse, it send a response to the source of the request. When a node receives a response, it is automatically forwarded to the client or the source of the request.

Local data structures

Data on each node is the following:

- Configuration and list of neighbors loaded from a yaml file on startup;
- List of files in the warehouse built from a folder scan on startup;
- Hash map of the received query, key is the uuid of and value is the IP address to witch the responses must be sent;
- Hash map of the received client requests, key is the uuid of the request and value is the IP address of the client.

Client

At startup, the client sends its request to a node of the network. It then collects responses during a certain time before showing the user all the corresponding files. When sending a request, the client automatically generates an uuid using the google Go implementation of RFC4122 that guaranties the uniqueness of the generated string across space and time.

Local data structures

On the client, the only data is a map to store the responses to the query. Key is the filename and values are the IP addresses of all the nodes where the file is available.

Message structure

A message is separated in multiple fields by a semicolon. The first letter indicates the type of the message.

Q messages

These are the messages exchanged between nodes containing queries. Fields are:

- Message type: Q
- UUID of the request, generated by the client and never changes
- Searched terms
- TTL, initialized at 5, it is decremented each time the request is received by a node. When equal to 0, the request isn't processed anymore.
- Source IP, each node sending this type of message puts in own IP in.

Example: `Q;f404ff9a-ea04-477e-bea7-8f14931538cc;debian;4;127.0.1.2`

R messages

These are the response messages, they are going through the network to the client when a match is found for a request. Fields are:

- UUID, generated by the client and never changed
- Complete name file, as found in the warehouse
- Ip address of the node containing the file

Example: `R;f404ff9a-ea04-477e-bea7-8f14931538cc;Debian-Desktop-11.iso;127.0.1.6`

C message

This is the message created by a client to make a request. Fields are:

- UUID, generated by the client, unique across all the network
- Searched term
- Ip of the client

Example: `C;f404ff9a-ea04-477e-bea7-8f14931538cc;debian;127.0.2.1`