Gnutella Web Caching System

Table of Contents

1. Introduction
2. Client-Script Interface
    1. "Hostfile" Request
    2. "Urlfile" Request
    3. Update Request
    4. Ping Request
    5. Other Requests / Extensions
    6. Statistics
    7. Client Version Information
3. Client-Side Behavior
    1. Requests
4. Server-Side Behavior
    1. Security
5. Change Log

1. Introduction

(ripped with many thanks from Info Anarchy's summary)
The goal of the "Gnutella Web Caching System" (the "cache") is to eliminate the "Initial Connection Point Problem" of a fully decentralized network: Where do I find a first host to connect to? The cache is a program (script) placed on any web server that stores IP addresses of hosts in the Gnutella network and URLs of other caches. Gnutella clients connect to a cache in their list randomly. They send and receive IP addresses and URLs from the cache. With the randomized connection it is to be assured that all caches eventually learn about each other, and that all caches have relatively fresh hosts and URLs. The concept is independent from Gnutella clients.

These specifications exist to define the interface between a Gnutella client and a web cache script. They also include many descriptions as to how a client and a cache script should behave. While these descriptions are not necessarily set in stone, they are designed to provide optimal interaction between clients and scripts and should therefore be implemented. *All* developers should take care that the interactions as described here are followed and should never release scripts or clients without proper in-house testing first, as to not disrupt the integrity of the network.

Client Developers: The cache system gains its strength through having many caches spread on many web servers. If you include the GWebCache functionality in your client, please make sure to also advertise the GWebCache scripts that are available!

^ Top ^

2. Client-Script Interface

2.1. "Hostfile" Request

The client wishes to receive a list of Gnutella nodes.

Request: `URL?hostfile=1`

Response: A return-separated list of Gnutella nodes in the format "ip:port" (numerical IPs only). The list should not be very long (around 20 nodes) and should contain only the newest entries. (Returning an empty list is also possible in case the cache does not have any hosts stored yet.)
OR
A redirect (HTTP code 3xx) response, indicating that the client needs to send another HTTP GET request for the file. Clients must support this method. Luckily, many standard HTTP libraries automatically follow redirects. When a client follows the redirect, it should receive a list as described above.
OR
The string "`ERROR`", possibly followed by more specific error information.

2.2. "Urlfile" Request

Important Note: In all of the GWebCache system, URLs must always begin with http:// , in requests as well as responses.

The client wishes to receive a list of alternate web cache URLs.

Request: `URL?urlfile=1`

Response: A return-separated list of alternate web caches' URLs. The list should not be very long (around 10-20 URLs) and should contain only the newest entries. (Returning an empty list is also possible in case the cache does not have any URLs stored yet.)
OR
A redirect (HTTP code 3xx) response, indicating that the client needs to send another HTTP GET request for the file. Clients must support this method. Luckily, many standard HTTP libraries automatically follow redirects. When a client follows the redirect, it should receive a list as described above.
OR
The string "`ERROR`", possibly followed by more specific error information.

2.3. Update Request

The client wishes to update IP addresses and/or alternate web cache URLs to a cache.

Request: `URL?ip=XXX.XXX.XXX.XXX:PORT` *OR*
`URL?url=http://WWW.SOMEHOST.COM/PATH/TO/SCRIPT` *OR*
`URL?ip=XXX.XXX.XXX.XXX:PORT&url=http://WWW.SOMEHOST.COM/PATH/TO/SCRIPT`
Reminder: Requests need to be URL-Encoded - see "Requests" in "Client-Side Behavior"
For compatibility issues, scripts should also accept `ip1` instead of `ip` and `url1` instead of `url`. Once older clients that still use `ip1` and `url1` are phased out this will no longer be necessary.

Response: First line must be: either "`OK`" or "`ERROR`", or "`ERROR: Message`".
The following lines can be used by the script for further messages, such as information or warnings (can be ignored by the client).

2.4. Ping Request

A ping/pong scheme to verify that caches are active.

Request: `URL?ping=1`

Response: The first four characters of the response are: "`PONG`", followed by a version number string (can be omitted).

2.5. Other Requests / Extensions

Other requests that a script can implement include HTML information pages, statistics, etc. For example, if no request is sent to the script (i.e. the script is simply browsed to), it could display a page informing the user that "this is a Gnutella web cache" or something similar. Or, one could include an extra request, "`URL?stats=1`", which could display a HTML page with some statistics. In general, script authors can include any extensions they wish, as long as the interaction described above remains unchanged. The same thing is true for clients.

2.6. Statistics

Statistics are regularly collected on all known GWebCache scripts. If the author of a script would like to make statistics from their script available, the following request should be implemented.

Request: `URL?statfile=1`

Response: Line 1: Total number of requests received.
Line 2: Requests received in the last full hour.
Line 3: Update requests received in the last full hour. (*optional but strongly encouraged*)

2.7. Client Version Information

In order to collect statistics on client versions and identify possible problems with certain clients, clients should send along two extra parameters with every request: "`client`", which is the 4-character vendor identifier that is also used in query hits, and "`version`", which is the version number of that client.

Examples:
```
URL?client=GNUC&version=1.8.4.0&hostfile=1
URL?ip=XXX.XXX.XXX.XXX&client=BEAR&version=2.6.3
URL?client=LIME&version=2.4&ping=1
```

^ Top ^

3. Client-Side Behavior

Clients generally keep an internal cache of the IP addresses of known Gnutella nodes.

In addition to this list, they should also keep an internal list of web caches. Clients should keep track of whether the URLs in their caches are functional or not. The major issue is that when clients send Update requests to caches, they must send only the URLs of functional caches.

Specifically, "bad caches" are those that return:

- nothing - those that cannot be accessed at all (timeouts, invalid hostnames, etc.)
- HTTP error codes (400-599)
- responses that cannot be parsed by a client
- `ERROR` responses (more than a few times in a row)
- many non-functional hosts or URLs (*optional*)

When making requests, a client should pick a cache from its internal list - a different one every time.

A client should send a Hostfile request whenever it needs hosts to connect to.

A client should send a Urlfile request to build its internal list of caches (such as once on start up).

Clients should only send updates if they accept incoming connections - i.e. clients behind firewalls should not send updates. Also, if supported by clients, only Ultrapeers/Supernodes should send updates. After a client has been up for an hour, it should begin sending an Update request periodically - every 60 minutes. It sends its own IP address and port in the "`ip`" parameter and a the URL of a random cache in the "`url`" parameter. Clients should only submit the URLs of caches that they know are functional!

The Ping request can and should be used to verify that a URL is valid and that a script is functioning correctly. Note: Some scripts, when installed by users on their servers, may return `PONG`s correctly but fail on other requests (mostly due to file access errors and the like). However, if a script responds correctly to an Update request, it is most likely working well. So if a script responds correctly to *both* a Ping and an Update correctly it can be assumed to be functional.

3.1. Requests

Interaction with the web server and cache is a series of HTTP GET requests and responses. Support for POST requests is optional and not necessary. In these specifications, notation `URL?query` indicates the URLs of a script with the attached query string, where "`query`" is a series of `name=value` pairs joined by "`&`"s. These name/value pairs must be "URL-Encoded", as is described (for example) here, or in RFC1738. Due to the differences between operating systems, responses can be LF, CRLF, or CR-terminated, but should be of Content-Type "text/*". Responses are interpreted line-by-line.

Tip: GET requests are easier than they may sound above: the query (the information/request you are sending the script) is simply part of the URL. For example, let's say the the request is: `URL?ip=192.168.0.1:123`, you will simply have to open the following URL using whatever web functions your programming language provides:
`http://www.somehost.com/path/to/script.php?ip=192.168.0.1:123`
The only tricky parts are: one, the "URL-Encoding" - your best bet is to go look for such functions, they have often already been written by someone and maybe already are part of your libraries. Second, interpreting the end-of-line characters in the responses - again, often there are already functions in the libraries that you can use to read responses line-by-line, taking the end-of-line characters into account.

^ Top ^

4. Server-Side (Script) Behavior

Scripts accept requests by clients through the standard HTTP GET mechanism and should respond according to these specifications, otherwise they risk being marked as invalid by a client. Responses should be of Content-Type "text/*" (i.e. any content-type that is text - such as text/plain or text/html).

An `OK` message usually means that everything went well and the script executed normally.

An `ERROR` message usually indicates some form of fatal error because of which the script could not do what is is supposed to. Since clients will (should) remove scripts that return error messages often, it is advised to return `ERROR`s only when the script is expected to be down for a while (such as, the script will be or has been removed from server, server overload, file errors, etc.).

In other words, things such as the submission of an invalid IP and/or URL, or even blocking clients according to the security measures described below, should more likely be answered by an `OK` followed by a warning message on the next line(s) instead of an `ERROR`. If this seems counter-intuitive, think of it this way: since clients will most likely handle the responses of scripts silently, you can think of an `ERROR` response to mean that the script is asking to be removed from the client's internal list of caches.

Scripts should only return a few (around 10-20) and only the newest Hosts and URLs. Therefore they only need to keep that many entries in their lists and can flush older entries as newer ones arrive. Keeping such a relatively small number of entries is essential to keeping all the information in all caches fresh.

4.1. Security

The most obvious attack of the cache system is attempting to submit invalid IP addresses and URLs. Luckily, the system is very error-tolerant: bad IP addresses and URLs will not break the system, only cause slight slowdowns because clients will have to try more URLs before they get a working IP. Also, because scripts are constantly receiving updates from many clients, bad entries in the caches will be flushed out within a short period of time.

Caches should implement the following simple security feature: Once a client has made an Update request, do not accept any more Update requests from that client's IP address for another ~55 minutes (according to the update frequency of the clients - 60 minutes). That way, people attempting to submit bad entries to a cache will only be able to submit once an hour, while their entries will only last in the caches for a fraction of an hour because other clients are constantly submitting valid IPs and URLs.

Note: Scripts should still accept Hostfile and Urlfile requests, since these do not change the data that the cache is storing.

Scripts can check the validity of the submitted IP address by verifying that it is the same as the one that is making the request - in the Apache CGI environment (and probably other servers' environments too) this information is passed in the "`REMOTE_ADDR`" environment variable.

Scripts may whish to check the validity of submitted URLs by sending a Ping request, but this is not required.

^ Top ^

5. Change Log

v1.3.1
- Added TOC

v1.3
- Added client version extension

v1.2
- **Changed Update Request format**
- Updated Statistics response
- Reorganized document
- Added security information

v1.1
- Suggested client and server-side behavior more specific.
- Added suggested statistics response.

v1.0
- First release.

^ Top ^

GWebCache Home
See also: http://www.gnucleus.com/