Metalink

Try not to download the same file twice. Improve cache efficiency and speed up downloads.

Take standard headers and knowledge about objects in the cache and potentially rewrite those headers so that a client will use a URL that's already cached instead of one that isn't. The headers are specified in RFC 6249 (Metalink/HTTP: Mirrors and Hashes) and RFC 3230 (Instance Digests in HTTP) and are sent by various download redirectors or content distribution networks.

Who Cares?

More important than saving a little bandwidth, this saves users from frustration.

A lot of download sites distribute the same files from many different mirrors and users don't know which mirrors are already cached. These sites often present users with a simple download button, but the button doesn't predictably access the same mirror, or a mirror that's already cached. To users it seems like the download works sometimes (takes seconds) and not others (takes hours), which is frustrating.

An extreme example of this happens when users share a limited, possibly unreliable Internet connection, as is common in parts of Africa for example.

How to cache openSUSE repositories with Squid is another, different example of a use case where picking a URL that's already cached is valuable.

What It Does

When it sees a response with a "Location: ..." header and a "Digest: SHA-256=..." header, it checks if the URL in the Location header is already cached. If it isn't, then it tries to find a URL that is cached to use instead. It looks in the cache for some object that matches the digest in the Digest header and if it succeeds, then it rewrites the Location header with that object's URL.

This way a client should get sent to a URL that's already cached and won't download the file again.

How to Use It

Just build the plugin and add it to your plugin.config file.

The code is distributed along with recent versions of Traffic Server, in the plugins/experimental/metalink directory. To build it, pass the --enable-experimental-plugins option to the configure script when you build Traffic Server:

```
```

When you're done building Traffic Server, add "metalink.so" to your plugin.config file to start using the plugin.

Status of the Code

It implements `TS_HTTP_SEND_RESPONSE_HDR_HOOK` to check and potentially rewrite the Location and Digest headers after responses are cached. It doesn't do it before they're cached because the contents of the cache can change after responses are cached. It uses `TSCacheRead()` to check if the URL in the Location header is already cached. In future, the plugin should also check if the URL is fresh or not.

It implements `TS_HTTP_READ_RESPONSE_HDR_HOOK` and a null transformation to compute the SHA-256 digest for content as it's added to the cache. It uses `SHA256_Init()`, `SHA256_Update()`, and `SHA256_Final()` from OpenSSL to compute the digest, then it uses `TSCacheWrite()` to associate the digest with the request URL. This adds a new cache object where the key is the digest and the object is the request URL.

To check if the cache already contains content that matches a digest, the plugin must call `TSCacheRead()` with the digest as the key, read the URL stored in the resultant object, and then call `TSCacheRead()` again with this URL as the key. This is probably inefficient and should be improved.

An early version of the plugin scanned "Link: <...>; rel=duplicate" headers. If the URL in the "Location: ..." header wasn't already cached, it scanned "Link: <...>; rel=duplicate" headers for a URL that was. The "Digest: SHA-256=..." header is superior because it will find content that already exists in the cache in every case that a "Link: <...>; rel=duplicate" header would, plus in cases where the URL is not listed among the "Link: <...>; rel=duplicate" headers, maybe because the content was downloaded from a URL not participating in the content distribution network, or maybe because there are too many mirrors to list in "Link: <...>; rel=duplicate" headers.

The "Digest: SHA-256=..." header is also more efficient than "Link: <...>; rel=duplicate" headers because it involves a constant number of cache lookups. RFC 6249 requires a "Digest: SHA-256=..." header or "Link: <...>; rel=duplicate" headers MUST be ignored:

> *If Instance Digests are not provided by the Metalink servers, the Link header fields pertaining to this specification MUST be ignored.*

> *Metalinks contain whole file hashes as described in Section 6, and MUST include SHA-256, as specified in [FIPS-180-3].*

Alex Rousskov pointed out a project for Squid to implement Duplicate Transfer Detection:

- http://thread.gmane.org/gmane.comp.web.squid.devel/15803
- http://thread.gmane.org/gmane.comp.web.squid.devel/16335
- http://www.hpl.hp.com/techreports/2004/HPL-2004-29.pdf

Per Jessen is working on another project for Squid with a similar goal: http://wiki.jessen.ch/index/How_to_cache_openSUSE_repositories_with_Squid