Frontier client retry strategy

This section only describes frontier client version 2.8.15 (February 2016) and later. See below for the differences in older client versions. For comparison you could see the CVMFS network path selection description.

Definitions:

- **proxy group** - either all the IP addresses of one IP family (IPv4 or IPv6) in a round-robin DNS name in a proxyurl or a backupproxyurl, or all the proxyurls together if loadbalance=proxies is set.

If any connections are already established and working, the frontier_client will continue to reuse that connection for subsequent queries until one of them returns an error or a response is larger than 16KB. Squid also drops connections from its end after they have been idle for 1 minute. When connections are dropped in one of these ways, the next query will attempt to reconnect to a proxy within the same proxy group that had not previously generated an error. Using a different proxy in the proxy group when possible is done to improve load balancing. The server URL will be the same unless loadbalance=servers is set, in which case another random server that hasn't had a server error will be chosen.

When any query needs to be done and more than 5 minutes has elapsed since the first connection, the connection is also dropped, the proxy list is reset to the beginning, all cached DNS names of proxies are considered to be invalid so they will need to be looked up again the next time they are referenced, all proxy errors are cleared so they will be retried, and the starting time of the first connection is reset so the same process can be done 5 minutes later. This is all in case a proxy that was having a problem had since been fixed. If it has been more than 30 minutes since the first connection (using a separate starting time variable) then the server list is also reset, server errors errors are cleared, and that starting time is reset. The reason for the longer time for servers is because of some of the details below (servers are tried with every proxy in a group, and read timeouts are longer, so for example a dead server tried with 4 proxies can take 40 seconds, which is a significant percentage of 5 minutes).

Since there are usually many combinations of proxies and servers to try, it is important that the client does not spend much time on any one combination. For that reason, the default connect timeout time is 5 seconds and the default read timeout time is 10 seconds.

Order of proxy groups:

- In general, proxy groups are tried in the order they appear in the configuration. The exceptions are:
  - if loadbalance=proxies is set, the proxyurls are randomized
  - backuproxyurls always come after proxyurls
- If there are both IPv4 and IPv6 addresses in the same DNS name (either proxyurl or backuprpoxyurl), the IP family chosen by the preferipfamily option (default 4) will be the first group and the other family will be the next group. If preferipfamily=0, the family of the first address returned by the operating system will be the preferred family.

The frontier client error retry strategy distinguishes between five types of non-fatal errors:

1. A cache **max-age exceeded condition** -- the server specified a maximum cache age in its response, and the received "Age" http header (inserted by the squid proxy) is greater than that. The server sets this maximum age near the end of its response when it is too late to put it in the http header, which happens when a condition to limit the max age is found after the http header has already been sent by the server. The http header is sent either when a buffer full of data is ready or when a "keepalive" has to be sent because the server has been waiting for more than 5 seconds for that client's request to the database. A condition that the server might notice at that point is either some kind of database error or an empty result. If the server doesn't set the maximum cache age but it is a protocol error (see below),

then the client uses a default maximum cache age of 5 minutes (this is to support servlet versions older than 3.33).
2. A **server error** -- a proxy or server responded but there was either an http error code from the server where it knew it had a problem, or the proxy immediately determined a server problem and returned an error code indicating so. In other words, there was an http error code of either 404 or in the 500 series. The result will not be cached. Server errors can also be caused by a variety of problems with downloading the host certificate when the signing security feature is enabled, and that result may be cached.
3. A **connect error** - the socket couldn't be connected, meaning that a proxy or direct-connect server is down. This can be connection timeout, connection refused, or network unreachable.
4. A **protocol error** - either the response http code was not 200 OK (and the code was not 404 or in the 500 server-error series, covered above), the response couldn't be completely parsed as expected, or the server signaled at the end of its response that it encountered a late error and should not be cached for a long time (even though the http headers said it was OK and could be cached).
5. All other types of errors, for example networking problems, read timeouts because of overloading or non-response from a server, etc.

These are the retry strategies:

1. If there is a max-age exceeded condition, retry the same server and proxy with the specified maximum cache age N seconds (Cache-control: max-age=N). If the condition persists after the retry (should only happen on buggy versions of squid), ignore it and move on to the next strategy.
2. If a server error happens, keep the same proxy and try the next server URL. This server is flagged as having had an error so it is not tried again until after a 30 minute reset (this is mainly relevant when loadbalance=servers). However, if this happens on the last server, clear all the server error flags, go back to the beginning of the server list and try the next proxy with the first server (in case a proxy was somehow returning a server error).
3. If a connect error to a proxy happens, try the next proxy, including within the same proxy group if there are any left that hadn't had an error, with the same server URL. This proxy is flagged as having had an error so it is not tried again until after a 5 minute reset or after there are no more proxies.
4. If a protocol error happens **and** the soft retry of step 1 has already happened (which means there was a max-age exceeded condition), with the same proxy & server retry the request with a hard refresh (Pragma: no-cache). (The hard refresh is needed to clear the cache in cases that a soft refresh doesn't clear because of If-Modified-Since revalidation.) If that still returns a non-server error, try the next proxy with the same server as long as the proxy is in the same proxy group. If the end of the proxy group is reached, start again at the beginning of the proxy group and try the next server until there are no more servers. If there are no more servers, clear any server error flags, go back again to the beginning of the server list, and move to the next proxy group.
5. For all other errors, do the same as a protocol error except without the hard refresh.
6. When there are no more proxies to try, unless failovertoserver=no is set (and backupproxyurl implies that), try directly connecting to all servers including all the addresses in each round-robin DNS name. Note: this is the only time that all the addresses in a round-robin DNS server name are used, since when going through a proxy only the server URL is used, not individual IP addresses. So if it is important to guarantee that all servers be tried through proxies, the individual server names need to be included in the list of serverurls (typically after the round-robin name).

Since the first strategy resets the server list while others reset to the beginning of a proxy group, and since each try can independently return its own error, it is possible to alternate between the strategies and never terminate nor make progress. For example, if there are two proxies in the same group PROXY1 and PROXY2 and two servers SERVER1 and SERVER2, and SERVER1 is totally down (which translates to a read timeout, the last type of error) while SERVER2 always returns a server error, then the last two steps will keep repeating:

- PROXY1+SERVER1: read timeout, advance proxy

- PROXY2+SERVER1: read timeout, go back to beginning of proxy group with next server
- PROXY1+SERVER2: server error, no more servers so go back to the beginning of the server list and try next proxy

To avoid that situation, if the server list has been set back to its beginning by a server error while in a given proxy group since the last 5-minute full reset, do not try to start again at the beginning of that proxy group even if the strategy calls for it.

Older client version differences

Frontier clients prior to version 2.8.2 (June 2011) are not documented on this page.

For frontier clients 2.8.9 (January 2014) through 2.8.14, these are the differences compared to 2.8.15 (February 2016) above:

1. Different IP families in the same DNS name were not considered separate groups (in fact IPv6 was not supported at all before 2.8.14).
2. 404 was considered a protocol error, not a server error.
3. Connection refused and network unreachable were not considered connect errors, only connection timeout was.
4. There was a bug present since 2.8.6 that considered a whole proxy group as failed when the last one failed, even if previous ones hadn't failed. That could happen after load balancing within a proxy group.
5. There was a bug present since 2.8.9 where if a server did not send an Age header (because it was a fresh query) and the previous query on the same connection had an old enough Age, a max-age exceeded condition could be erroneously triggered; the Age from the previous query was applied to the one without an Age.

For frontier clients version 2.8.7 (June 2013) and 2.8.8, there are these differences compared to the 2.8.9 strategies above:

1. There was no concept of a max-age exceeded condition.
2. For a protocol error, the request would be immediately retried first with a soft retry (Cache-control: max-age=0) before the hard retry.

For frontier client version 2.8.6 (April 2013) compared to the 2.8.7 strategies above:

- There was no digital signature support so there were no server errors with downloading host certificates.

For frontier client versions 2.8.2 through 2.8.5, these are the additional differences:

1. The entire proxy (and server) list and DNS name lookups were not reset at once after 5 (and 30) minutes. Instead, if any DNS name of a proxy or server (including round-robin DNS names) had not been looked up in at least 5 minutes and was attempted to be used again, that name was re-looked up in the DNS and all errors on the addresses in that name were considered cleared. Also, on every new connection it would go back to the beginning of the proxy & server list and look for one of each that hadn't been implicated in an error. The net result for proxies is not a lot different, other than being harder to explain. The net result for server errors is different because when going through proxies, server names are not looked up in the DNS so those errors never got cleared.
2. A proxy connect timeout was not treated any differently from other network errors.
3. For non-server errors, the next proxy in the list was always tried using the same server until running out of proxies, then the proxy list was reset and the next server was used. The different addresses in the same proxy round-robin DNS name were not all tried; if one of them had an error, it would move

on to the next proxy. (On the other hand, with loadbalance=proxies all the listed proxyurls were tried, as is still the case).

4. For direct connections to servers, similar to proxies only one address in round-robin DNS names were tried when there were errors.

This topic: Frontier > ClientRetryStrategy
Topic revision: r30 - 2018-03-09 - DaveDykstra