Age Header Field in HTTP/1.1

1. Problem Statement

HTTP/1.1 [1] defines the Age header field in section 14.6:

   The Age response-header field conveys the sender’s estimate of the amount of time since the response (or its revalidation) was generated at the origin server. A cached response is "fresh" if its age does not exceed its freshness lifetime. Age values are calculated as specified in section 13.2.3.

   Age = "Age" ":" age-value

   age-value = delta-seconds

Age values are non-negative decimal integers, representing time in seconds.
If a cache receives a value larger than the largest positive integer it can represent, or if any of its age calculations overflows, it MUST transmit an Age header with a value of 2147483648 \(2^{31}\). HTTP/1.1 caches MUST send an Age header in every response. Caches SHOULD use an arithmetic type of at least 31 bits of range.

This document focuses on the ambiguous use of the term "caches" in the second-to-last line above. The ambiguity is due to the fact that a cache never sends responses --- only a server application (proxy, gateway, or origin server), which may or may not include a cache, is capable of sending a response. HTTP/1.1 defines a "cache" as

A program’s local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cachable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server that is acting as a tunnel.

There are two possible interpretations of

HTTP/1.1 caches MUST send an Age header in every response.

Either

a) An HTTP/1.1 server that includes a cache MUST send an Age header field in every response.

or

b) An HTTP/1.1 server that includes a cache MUST include an Age header field in every response generated from its own cache.

The remainder of this document discusses the relative merits of these two options, referred to as "Option A" and "Option B", concluding in section 5 with a set of proposed changes to remove the ambiguity from future editions of the HTTP/1.1 specification.

2. Review of HTTP/1.1 Response Age Calculation

HTTP/1.1 defines an algorithm for calculating the age of a response message upon receipt by a cache. This document does not propose any modification of this algorithm; we describe it here in order to provide the background necessary to understand the later analyses. We only provide a brief summary here -- for a full explanation, see section 13.2.3 (Age Calculations) of RFC 2068 [1].

Summary of age calculation algorithm, when a cache receives a response:

```c
/*
 * age_value
 * is the value of Age: header received by the cache with
 * this response.
 * date_value
 * is the value of the origin server’s Date: header
 * request_time
 * is the (local) time when the cache made the request
 * that resulted in this cached response
 * response_time
 * is the (local) time when the cache received the
 * response
 * now
```
apparent_age = max(0, response_time - date_value);
corrected_received_age = max(apparent_age, age_value);
response_delay = response_time - request_time;
corrected_initial_age = corrected_received_age + response_delay;
resident_time = now - response_time;
current_age = corrected_initial_age + resident_time;

3. Analysis of Option A

If we were to assume that an HTTP/1.1 server that includes a cache MUST send an Age header field in every response.

is true, then an HTTP/1.1 proxy containing a cache would be required to add an Age header field value to every response that was forwarded, including those that were obtained first-hand from the origin server and never touched by the caching mechanism. This would directly contradict the paragraph in section 13.2.1 of RFC 2068 that states:

The expiration mechanism applies only to responses taken from a cache and not to first-hand responses forwarded immediately to the requesting client.

and also directly contradicts the last paragraph of section 13.2.3 of RFC 2068 that states:

Note that a client cannot reliably tell that a response is first-hand, but the presence of an Age header indicates that a response is definitely not first-hand.

If we further assume that the above two paragraphs are in error, then the following example illustrates the effect of the age calculation when a first-hand response passes through a hierarchical system of proxy caches (A, B, C), with each segment taking (a, b, c, d) amount of time to satisfy the request:

UA -------> A -------> B ---------> C -------> OS
              a            b             c             d

Since the age calculation includes an estimation of clock skew by each recipient (apparent_age), we also have the variables

skewC = max(0, response_time(C) - date_value(OS));
skewB = max(0, response_time(B) - date_value(OS));
skewA = max(0, response_time(A) - date_value(OS));
skewUA = max(0, response_time(UA) - date_value(OS));

then the received age will be calculated as follows:

At C:  age=max(skewC,0)+d
B:  age=max(skwB,max(skewC,0)+d)+(c+d)
A:  age=max(skwA,max(skwB,max(skewC,0)+d)+(c+d))+(b+c+d)
UA:  age=max(skwUA,max(skwA,max(skwB,max(skewC,0)+d)+(c+d))+(b+c+d))+(a+b+c+d)

Because the response is first-hand, we know that the real age at UA must be less than (a+b+c+d). Note that (a+b+c+d) will always be
added by UA, so the cumulative overestimation of the age will be at least

$$\max(\text{skewUA}, \max(\text{skewA}, \max(\text{skewB}, \max(\text{skewC}, 0) + d) + (c+d)) + (b+c+d))$$

If we further assume that all clocks are synchronized (the minimum case), then the age at UA will be estimated as

$$d + (c+d) + (b+c+d) + (a+b+c+d)$$

Note that the above is the minimum overestimation; since the variables skewC, skewB, skewA, and skewUA are all unbounded, the clock skew of each host on the request path adds to the perceived response age of all downstream recipients. Furthermore, a fast clock on the origin will add to the overestimated age at each hop.

However, in section 13.2.3 of RFC 2068, we also find

In essence, the Age value is the sum of the time that the response has been resident in each of the caches along the path from the origin server, plus the amount of time it has been in transit along network paths.

which in our example would imply an age value of \((a+b+c+d)\). Thus, Option A would result in an incorrect calculation of the age value, resulting in an overestimation of age in all cases, with the amount of error bounded only by the synchronization of clocks for each and every recipient along the request chain, plus the cumulative overestimation of the network transit time by each recipient.

4. Analysis of Option B

If we were to assume that

An HTTP/1.1 server that includes a cache MUST include an Age header field in every response generated from its own cache.

then an Age header field would not be added to a response that is received first-hand, and thus we would not contradict the sections of RFC 2068 that were quoted above.

Using the same example as in the analysis of Option A, the calculation of age with Option B would be as follows:

At C:  \(\text{age}=\max(\text{skewC}, 0) + d\)
B:  \(\text{age}=\max(\text{skewB}, 0) + (c+d)\)
A:  \(\text{age}=\max(\text{skewA}, 0) + (b+c+d)\)
UA:  \(\text{age}=\max(\text{skewUA}, 0) + (a+b+c+d)\)

Note that there is no cumulative overestimation of the age. The estimated age value at each recipient is only dependent on the skew between the recipient’s clock and that of the origin server, plus the total amount of time the request and response has been in transit along the network path. The minimum estimated age at UA is

\((a+b+c+d)\)

which matches the description provided in section 13.2.3 of RFC 2068.

5. Counter-arguments
The only argument voiced against Option B is that the calculation is "less conservative" than Option A, and that being "conservative" is better in order to "reduce as much as possible the probability of inadvertently delivering a stale response to a user."

If "conservative" means "always overestimates more than the other option", then the argument is certainly true. However, if the purpose of Age was to provide an overestimate, then why stop there? Why not add arbitrary amounts of age to forwarded response, just in case? Why not disable caching entirely?

The reason is because HTTP caching is good for the Internet as a whole, and in particular for the owners of the network bandwidth that would be used to satisfy a request that has already been cached. Overestimating response age reduces the effectiveness of caching, and thus results in increased network congestion, added bandwidth requirements, and in some cases additional per-packet charges.

Age was created to compensate for the possibility that clock skew between the origin server (represented by the Date header field) and the user agent (represented by the request time) might result in the age of a response being underestimated. Age was created so that HTTP/1.1 caches can communicate the actual observed age, thus providing a lower-bound for the age calculation that would be more reliable than simply calculating the difference between the date stamps.

If Age is to be useful, it must be trusted by cache implementers. In order to be trusted by cache implementers, the value of the Age header field must match its definition: the age of the response as observed by the application that generated the response message.

Furthermore, Option B is guaranteed to be conservative if all of the applications involved are HTTP/1.1-compliant or if the recipient’s clock is equal to or ahead of the origin server clock. The only case in which Option A *might* result in a better estimation than Option B is where one or more HTTP/1.0 caches are in the request chain AND the response came from one of those HTTP/1.0 caches in which it resided for some time AND the user agent’s system clock is running behind the origin server’s clock. In this one case, Option A would compensate for the clock skew if there existed an HTTP/1.1 cache between the user agent and the HTTP/1.0 cache generating the response AND the HTTP/1.1 cache is better-synchronized to the origin server clock.

The above scenario would require a minimum of two proxies in the chain, with at least one outer proxy being an old HTTP/1.0 cache and at least one inner proxy using HTTP/1.1. Given that, for many other reasons (described in RFC 2068), an HTTP/1.0 proxy is incapable of reliably caching HTTP messages in a proxy hierarchy, this scenario is not compelling.

In contrast, Option A would overestimate the age on all HTTP/1.1 requests, even when there are no longer any HTTP/1.0 proxies. It would also make the age calculation dependent on the clock synchronization of every recipient along the request chain, with the possibility for drastic overestimation if any of the recipients has a bad clock. Option A would therefore make the Age header field value consistently less reliable than simple comparison of date stamps.

5. Conclusion and Proposed Changes
Option B is the correct interpretation of when the Age header field should be added to an HTTP/1.1 response. The following changes to RFC 2068 will remove the ambiguity.

In section 14.6 (Age), replace the sentence

HTTP/1.1 caches MUST send an Age header in every response.

with

An HTTP/1.1 server that includes a cache MUST include an Age header field in every response generated from its own cache.

In section 13.2.3 (Age Calculations), replace the paragraph

HTTP/1.1 uses the Age response-header to help convey age information between caches. The Age header value is the sender’s estimate of the amount of time since the response was generated at the origin server. In the case of a cached response that has been revalidated with the origin server, the Age value is based on the time of revalidation, not of the original response.

with

HTTP/1.1 uses the Age response-header to convey the estimated age of the response message when obtained from a cache. The Age field value is the cache’s estimate of the amount of time since the response was generated or revalidated by the origin server.

Delete the following paragraph from section 13.2.3:

Note that this correction is applied at each HTTP/1.1 cache along the path, so that if there is an HTTP/1.0 cache in the path, the correct received age is computed as long as the receiving cache’s clock is nearly in sync. We don’t need end-to-end clock synchronization (although it is good to have), and there is no explicit clock synchronization step.

Replace the following two paragraphs from section 13.2.3:

When a cache sends a response, it must add to the corrected_initial_age the amount of time that the response was resident locally. It must then transmit this total age, using the Age header, to the next recipient cache.

Note that a client cannot reliably tell that a response is first-hand, but the presence of an Age header indicates that a response is definitely not first-hand. Also, if the Date in a response is earlier than the client’s local request time, the response is probably not first-hand (in the absence of serious clock skew).

with

The current_age of a cache entry is calculated by adding the amount of time (in seconds) since the cache entry was last validated by the origin server to the corrected_initial_age. When a response is generated from a cache entry, the server must include a single Age header field in the response with a value equal to the cache entry’s current_age.

The presence of an Age header field in a response implies that a
response is not first-hand. However, the converse is not true, since the lack of an Age header field in a response does not imply that the response is first-hand unless all caches along the request path are compliant with HTTP/1.1 (i.e., older HTTP caches did not implement the Age header field).

6. Security Considerations

The proposed changes close a potential security problem with HTTP/1.1 which would become manifest if a proxy with a slow clock (due to a hardware malfunction, failure to properly set, or caused to be reset by some malevolent agent) adds an Age header field to every response it forwarded, instead of only to those retrieved from its own cache, and thus eliminating the ability of a compliant downstream cache to reduce bandwidth usage on a congested network. Although this is not a serious concern with today’s use of HTTP caching, future use of hierarchical cache networks would be impacted.

7. Acknowledgements

This document was derived from discussions by the author within the HTTP working group, particularly with Jeffrey C. Mogul.

9. References


9. Author’s Address

Roy T. Fielding
Department of Information and Computer Science
University of California, Irvine
Irvine, CA  92697-3425

Fax: +1(714)824-1715
EMail: fielding@ics.uci.edu