7.11.1 使用期和新鲜生存期

为了分辨已缓存文档是否足够新鲜，缓存只需要计算两个值：已缓存副本的*使用期*（age），和已缓存副本的*新鲜生存期*（freshness lifetime）。如果已缓存副本的时长小于新鲜生存期，就说明副本足够新鲜，可以使用。用 Perl 表示为：

```
$is_fresh_enough = ($age < $freshness_lifetime);
```

文档的使用期就是自从服务器将其发送出来（或者最后一次被服务器再验证）之后“老去”的总时间。[19]缓存可能不知道文档响应是来自上游缓存，还是来自服务器的，所以它不能假设文档是最新的。它必须根据显式的 Age 首部（优先），或者通过对服务器生成的 Date 首部的处理，来确定文档的使用期。

文档的新鲜生存期说明，已缓存副本在不能提供给客户端使用之前能够存在的时间长度。新鲜生存期考虑了文档的过期日期，以及客户端可能请求的任何新鲜度覆盖范围。

有些客户端可能愿意接受稍微有些过期的文档（使用 Cache-Control: max-stale 首部）。有些客户端可能无法接受会在近期过期的文档（使用 Cache-Control: min-fresh 首部）。缓存将服务器过期信息与客户端的新鲜度要求结合在一起，以确定最大的新鲜生存期。

7.11.2 使用期的计算

响应的使用期就是服务器发布响应（或服务器对其进行了再验证）之后经过的总时间。使用期包含了响应在因特网路由器和网关中游荡的时间，在中间节点缓存中存储的时间，以及响应在你的缓存中停留的时间。例 7-1 给出了使用期计算的伪代码。

例 7-1 HTTP/1.1 使用期计算算法计算了已缓存文档的总体使用期

```
$apparent_age = max(0, $time_got_response - $Date_header_value);
$corrected_apparent_age = max($apparent_age, $Age_header_value);
$response_delay_estimate = ($time_got_response - $time_issued_request);
$age_when_document_arrived_at_our_cache =
    $corrected_apparent_age + $response_delay_estimate;
$how_long_copy_has_been_in_our_cache = $current_time - $time_got_response;

$age = $age_when_document_arrived_at_our_cache +
    $how_long_copy_has_been_in_our_cache;
```

注 19：记住，服务器上总是有所有文档的最新版本的。