HTTP 使用期计算的细节有点儿棘手，但其基本概念很简单。响应到达缓存时，缓存可以通过查看 Date 首部或 Age 首部来判断响应已使用的时间。缓存还能记录下文档在本地缓存中的停留时间。把这些值加在一起，就是响应的总使用期。HTTP 用一些魔法对时钟偏差和网络时延进行了补偿，但基本计算非常简单：


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

缓存可以很方便地判断出已缓存副本已经在本地缓存了多长时间（这就是简单的簿记问题），但当响应抵达缓存时，要确定响应的使用期是比较困难的，因为不是所有服务器的时钟都是同步的，而且我们也不知道响应到过哪里。完善的使用期计算算法会试着对此进行补偿。

1. 表面使用期是基于Date首部的

如果所有的计算机都共享同样的、完全精确的时钟，已缓存文档的使用期就可以是文档的“表面使用期”——当前时间减去服务器发送文档的时间。服务器发送时间就是 Date 首部的值。最简单的起始时间计算可以直接使用表面时间：

```
$apparent_age = $time_got_response - $Date_header_value;
$age_when_document_arrived_at_our_cache = $apparent_age;
```

但并不是所有的时钟都实现了良好的同步。客户端和服务器时钟之间可能有数分钟的差别，如果时钟没有设置好的话，甚至会有数小时或数天的区别。[20]

Web 应用程序，尤其是缓存代理，要做好与时间值有很大差异的服务器进行交互的准备。这种问题被称为*时钟偏差*（clock skew）——两台计算机时钟设置的不同。由于时钟偏差的存在，表面使用期有时会不太准确，而且有时会是负的。

如果使用期是负的，就将其设置为零。我们还可以对表面使用期进行完整性检查，以确定它没有大得令人不可思议，不过，实际上，表面使用期可能并没错。我们可能在与一个将文档缓存了很久的父缓存对话（缓存可能还存储了原始的 Date 首部）：

```
$apparent_age = max(0, $time_got_response - $Date_header_value);
$age_when_document_arrived_at_our_cache = $apparent_age;
```

要明确 Date 首部描述的是原始服务器的日期。代理和缓存一定不能修改这个日期！

注 20：HTTP 规范建议客户端、服务器和代理使用 NTP 这样的时间同步协议来强制使用统一的时间基准。