# 缓存作用

* 减少了冗余数据的传输
* 降低了时延
* 减少了网络拥塞，节约了带宽。

# 缓存机制

**客户端：**

客户端可以在请求报文中通过以下首部进行缓存控制：

* cache-control：

1. no-store：代理应该尽快删除缓存副本，因为其中可能包含敏感信息
2. no-cache：除非缓存副本进行了再验证，否则客户端不会接受已缓存的资源。
3. max-age=<s>: 代理不能发送缓存时间长于<s>的缓存副本
4. min-fresh=<s>: 缓存副本必须至少在未来<s>依然保持新鲜，否则不能返回
5. max-stale： 代理可以随意提供过期的文件

**代理缓存**：

代理缓存收到请求时，对报文进行解析后进行匹配，如果没有匹配上，则转发请求给原始服务器。如果匹配上，再**检测缓存新鲜度**，如果仍然新鲜，则直接将缓存副本发给客户端。如果不新鲜，则向原始服务器发起**再验证过程**。

**服务**器：

服务器在发送响应时，通过下列首部来控制缓存：

* **cache-control**：

1. no-store：代理缓存转发报文后，立即删除
2. no-cache: 代理每次都必须发起再验证，无论缓存是否新鲜。**如果该指令伴随一个首部列表的话，那么内容在缓存之前必须先删除指定的首部**。
3. must-revalidate：缓存在没有跟原始服务器进行再验证的情况下，不能提供这个副本的**陈旧对象**。
4. **max-age**：设置失效时间，用法同expires，不过其是相对时间。更加精确。
5. public：响应可以被任何服务器缓存

* **expires**：缓存失效的时间（绝对时间），当代理缓存收到客户请求后，会进行缓存匹配，如果匹配上，则再对比缓存失效时间，如果仍然有效，那么代理直接将缓存副本发给客户。如果已过期，则**缓存向原始服务器发起再验证过程**。
* **last-modified:** 用于再认证过程
* **Etag：**实体标签，可用于再认证过程

**再验证过程**

当缓存文档不再新鲜时，代理需要向原始服务器发起再认证，代理通过下列首部来控制再认证过程：

* 条件GET：

1. if-modified-since：last-modifyed
2. if-none-matched：Etag

服务器收到再验证GET后，将：

1）条件验证通过：回复200和新的实体内容

2）条件验证未通过：回复304，并且不带实体主体

3）资源未找到，回复404

* **文档使用期计算**

1. 服务器在发送响应时，会带上Date首部，代理缓存收到响应后可以根据接收时间减去Date首部时间，得到一个**传输时间**，然后**传输时间**加上**已缓存时间**即为文档的使用时间。由于代理和服务器之间可能存在时钟不同步的问题，因此根据Date来计算文档使用时间可能会有较大误差。
2. Age首部：使用http 1.1的应用程序应该在Age首部值中加上文档在每个应用程序和网络传输过程中停留的时间。因此，age中记录的是一个相对时间。

* **新鲜度计算**

文档的新鲜度是综合服务器和客户端限制计算出来的。

# Cookie与缓存

缓存与cookie事务有关的文档要特别小心，因为这可能会导致用户隐私的泄露。缓存在处理cookie时应遵循以下原则：

1. 如果**响应中带set-cookie首部**，那么应该设置cache-Control：no-cache=“Set-cookie”和cache-control:must-revalidate，max-age=0。这样缓存中将不会缓存set-cookie首部，并且缓存收到请求时都会向原始服务器发送**再验证**，收到服务器的再验证响应后，缓存会将响应中的set-cookie添加到缓存副本中发送给客户。
2. **请求中带cookie首部**，这时得到的响应应该是针对特定用户的私有响应，因此，服务器应该将响应标记为不可缓存的，即设置cache-control：no-store首部。

# 缓存协议

* **ICP**：因特网缓存协议：如果某个缓存没有命中，可以向其兄弟缓存进行请求，从而避免查询原始服务器带来的开销。
* **CARP**：缓存阵列路由协议：通过URL应用散列函数，CARP就可以将此资源对象映射到特定的代理服务器上去。这样通过**单次查找**即可确定对象的位置。
* **HTCP**：超文本缓存协议
* **WCCP**：使路由器将Web流量重定向到代理缓存中去。WCCP负责路由器和缓存之间的通信，这样路由器可以

1. 对缓存进行验证（确保它们已启动且正在运行）
2. 在缓存之间进行**负载均衡**
3. 将特定类型的流量发送给特定的缓存。