同时从多条输入连接上读取数据，在整条报文抵达之前开始对事务进行处理。

7.7.2 第二步——解析

接下来，缓存将请求报文解析为片断，将首部的各个部分放入易于操作的数据结构
172 中。这样，缓存软件就更容易处理首部字段并修改它们了。[10]

7.7.3 第三步——查找

在第三步中，缓存获取了URL，查找本地副本。本地副本可能存储在内存、本地磁盘，甚至附近的另一台计算机中。专业级的缓存会使用快速算法来确定本地缓存中是否有某个对象。如果本地没有这个文档，它可以根据情形和配置，到原始服务器或父代理中去取，或者返回一条错误信息。

已缓存对象中包含了服务器响应主体和原始服务器响应首部，这样就会在缓存命中时返回正确的服务器首部。已缓存对象中还包含了一些*元数据*（metadata），用来记录对象在缓存中停留了多长时间，以及它被用过多少次等。[11]

7.7.4 第四步——新鲜度检测

HTTP通过缓存将服务器文档的副本保留一段时间。在这段时间里，都认为文档是
173 “新鲜的”，缓存可以在不联系服务器的情况下，直接提供该文档。但一旦已缓存副本停留的时间太长，超过了文档的*新鲜度限值*（freshness limit），就认为对象“过时”了，在提供该文档之前，缓存要再次与服务器进行确认，以查看文档是否发生了变化。客户端发送给缓存的所有请求首部自身都可以强制缓存进行再验证，或者完全避免验证，这使得事情变得更加复杂了。

HTTP有一组非常复杂的新鲜度检测规则，缓存产品支持的大量配置选项，以及与非HTTP新鲜度标准进行互通的需要则使问题变得更加严重了。本章其余的大部分篇幅都用于解释新鲜度的计算问题。

7.7.5 第五步——创建响应

我们希望缓存的响应看起来就像来自原始服务器的一样，缓存将已缓存的服务器响

注10：解析程序还要负责首部各部分的标准化，将大小写或可替换数据格式之类不太重要的区别都看作等效的。而且，某些请求报文中包含有完整的绝对URL，而其他一些请求中包含的则是相对URL和Host首部，所以解析程序通常都要将这些细节隐藏起来（参见2.3.1节）。

注11：复杂的缓存还会保留引发服务器响应的原始客户端响应首部的一份副本，以用于HTTP/1.1内容协商（参见第17章）。