Chapter 2
Background Information

Keywords Web request · Cacheable request · Web object · Web proxy servers · Squid

It is necessary to have a clear and precise definition on when a Web object is allowed to be cached. A thorough, well-defined definition allows Web cache users to understand what requests they make could potentially be cached, and as well, is necessary for system administrators as a good tool in improving quality of service (QoS) for end users. For this book, a *Web object* is a term used for all possible objects (HTML pages, images, videos, etc.) transferred through the HTTP protocol that can be stored in a proxy cache [1].

2.1 Web Request

A Web request is a reference made through the HTTP protocol to a Web object, primarily referenced by their uniform resource locator (URL). Requests are also identified by the size of the requested Web object from the origin server (at the time the request was made), a HTTP return code, and the time the proxy received the request. We define a *cacheable request* to have the following criteria:

- There must be a defined size, in bytes, for the request, and that size must be less than the total size of the cache and greater than zero.
- The request must be a GET request, and its status code must be one of the following, as set by the HTTP 1.1 protocol [2]: 200, 203, 206, 300, 301, or 410. Table 2.1 shows the status codes and their meanings.

Separate from *cacheable request*, we also ignore any requests with URLs containing "/cgi-bin/" as well as any URLs that are queries (those that contain a question mark in their URL after the last "/").

H. ElAarag, *Web Proxy Cache Replacement Strategies*,
SpringerBriefs in Computer Science, DOI: 10.1007/978-1-4471-4893-7_2,

Table 2.1 HTTP status code meanings

Code	Meaning
200	Ok
203	Non-authoritative information
206	Partial content
300	Multiple choices
301	Moved permanently
410	Gone (synonymous with deleted)

Once a request is known to be *cacheable* and is received by the proxy, several things will occur in a sequential order. In a basic proxy server model, if a *cache hit* occurs, then the object being referenced is in the cache and the data can be copied and sent to the client. On a *cache miss*, when no object in the cache matches the request, the Web object will be retrieved from the origin server and the *cache placement strategy* decides whether the object will be placed into the cache. If there is not enough room for the new object to be added, then the *cache replacement strategy* is invoked. However, in order to understand how these strategies work, we will define several aspects of objects these strategies will consider.

2.2 Characteristics of Web Objects

Web objects are identified by several different characteristics. Each replacement strategy requires usually a small subset of the characteristics; however, all Web objects must be identified by their URL, since this is the only unique factor. The most important characteristics of Web objects are as follows[1]:

- Recency: information relating to the time the object was last requested.
- Frequency counter: number of requests to the object.
- Size: the size, in bytes, of the Web object.
- Cost: the "cost" incurred for fetching the object from the origin server. Also known as the miss penalty.
- Request value: the benefit gained for storing the object in the cache. This is generally a heuristic-based on other characteristics, since an actual request value of an object cannot be determined without a priori information.
- Expiration time: Also generally a heuristic, defined either by the proxy or by the origin server of when the object will become stale and should be removed or refreshed in the cache. Also known as the time-to-live (TTL).

Most strategies use a combination of these characteristics to make their replacement decisions. The expiration time is the only characteristic mentioned that was not utilized in our simulation and is primarily referenced when dealing with the problem of *cache consistency*, which is out of the scope of this book. The request value is an abstract characteristic, primarily used by *function-based*

strategies, and defined by a characteristic function that pursues a total, well-defined ordering of the objects.[1]

2.3 Web Proxy Servers

Web proxy servers are generally configured in one of two basic architectures. In the first configuration, one or more proxy servers are positioned on the network between a Web server or group of Web servers and incoming client traffic from the WAN. The design is aimed at reducing load on the Web server(s). The second architecture is geared toward reducing network congestion. The proxy server(s) is located on the LAN and receives WAN-bound client requests. Figure 2.1 illustrates both architectures. The per-request behavior of a Web proxy server is independent of the choice in architecture. A request for a cached object is served directly. If a request is received for an object not in the cache, the proxy requests the object from the Web server, which potentially caches the object and then serves the object to the client. This process is shown in Fig. 2.2.

Web proxy caching is a paging problem. Strategies applied to other paging problems, such as main memory management, have been adapted to address Web proxy caching. Web proxy caching has two main factors that must be addressed: cache replacement and cache consistency. Cache replacement refers to deciding what should be removed from the cache to make room for new data. Cache consistency refers to ensuring that items in the cache are still the same as items on the original server. This book will address the former.

2.4 Squid

Squid [3] is an open source proxy cache project now maintained by volunteers, originally supported through an NSF grant, and now fueled purely by donations. Squid currently supports HTTP 1.0 protocol and is almost HTTP 1.1 compliant. One difference between HTTP 1.0 and 1.1 protocols is the cacheable request definition. For instance, partial requests (return code 206) are not allowed to be cached in Squid, as opposed to HTTP 1.1 compliant software, which is allowed to be cached. Through Squid's configuration files and easy compilation steps, one can control almost any aspect of the cache including the replacement strategy and how the proxy logs requests, such as keeping a log of the requests that Squid automatically saves to the hard disk. In the early 1990s, the Squid project came about

[1] Essentially, any characteristic could be the request value, if the algorithm makes its decision based on one variable that has a total, well-defined ordering.

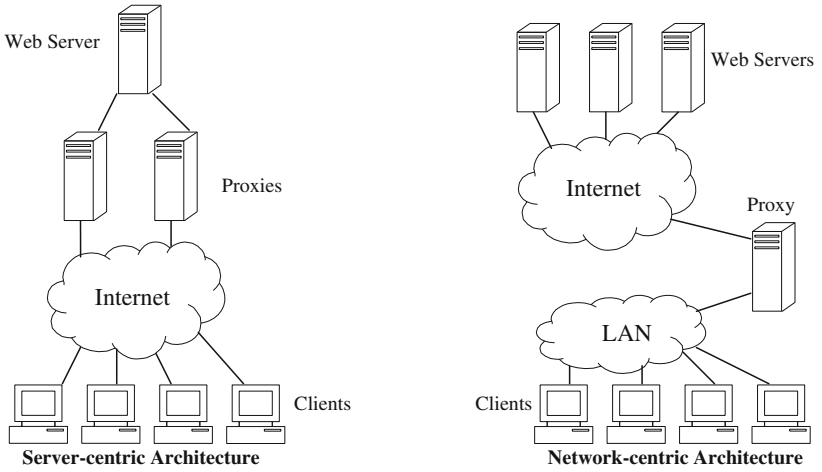

Fig. 2.1 Web proxy cache architectures

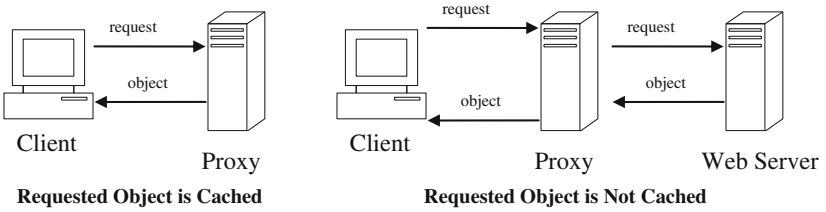

Fig. 2.2 Web proxy cache request sequence

from a fork of the Harvest Cache Daemon. The other fork existing today is NetApp's NetCache [3], which is a proprietary software package.

Originally, the NSF funded Squid in order to promote research in the 1990s before the conception of broadband, DSL, and other technologies that lead to a huge increase in bandwidth. As far as many businesses and researchers were concerned, the Web caching problem was essentially deemed "solved" in the late 1990s and early part of this decade [1]. Most of these strategies were simple due to the lack of CPU power, available RAM, and especially available storage. In fact, it was rare to see a proxy server that had more than 2 gigabytes of space devoted to the cache. Today, this amount of disk space is trivial.

However, the decrease in costs for bandwidth, increase in CPU power, greater storage capacities, and higher demand for download speed have led to bandwidth problems again especially for Web publishers who host major Web sites. Increase in use of dynamic pages as well is leading to yet another strain in the bandwidth provided by today's broadband/WAN technologies. These new dilemmas have led many companies, businesses, and even Internet service providers (ISPs) to turn to

research in caching. However, many strategies authored in the late 1990s and early part of this decade are failing in being able to dynamically adapt to request streams and the constantly changing environment that the Internet has now become.

References

1. S. Podlipnig, L. Boszormenyi, A survey of web cache replacement strategies. ACM Comput. Surveys **35**(4), 374–398 (2003)
2. Hypertext transfer protocol–HTTP/1.1 [online document] [cited Aug. 14, 2007] available. WWW: http://www.w3.org/Protocols/rfc2616/rfc2616.html
3. What is squid?, available at http://www.squid-cache.org/Intro/

http://www.springer.com/978-1-4471-4892-0

Web Proxy Cache Replacement Strategies
Simulation, Implementation, and Performance Evaluation
ElAarag, H.
2013, X, 103 p. 81 illus., 15 illus. in color., Softcover
ISBN: 978-1-4471-4892-0