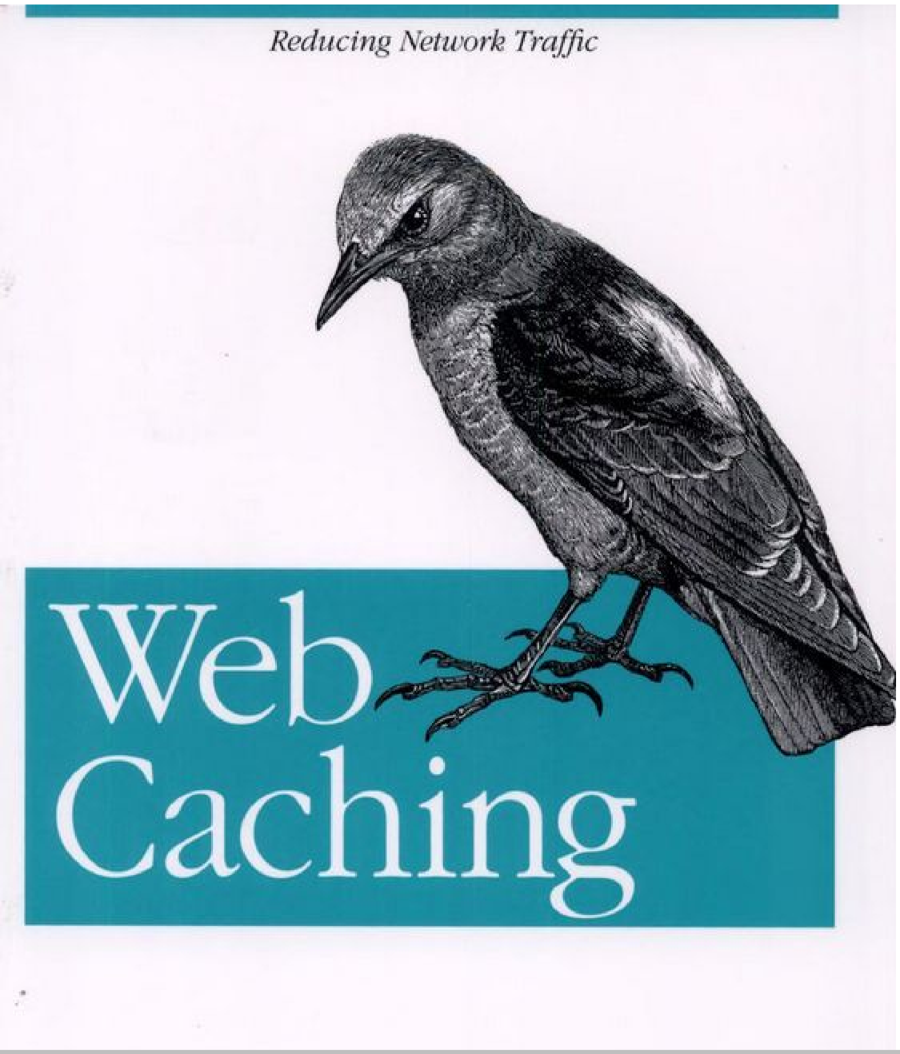

Reducing Network Traffic

Web Caching

Web Caching

Duane Wessels

Web Caching
by Duane Wessels

Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

Editors: Nathan Torkington and Paula Ferguson

Production Editor: Leanne Clarke Soylemez

Cover Designer: Edie Freedman

Printing History:

June 2001: First Edition.

Library of Congress Cataloging-in-Publication Data

Wessels, Duane.
Web Caching/Duane Wessels
p. cm.
ISBN 1-56592-536-X
1. Cache memory. 2. Browsers (Computer programs) 3. Software configuration management. 4. World Wide Web. I. Title.

TK7895.M4 W45 2001
004.5'3--dc21 2001033173

R_n
: Average ICP service time from neighbor caches

P
: ICP hit ratio

The average service time without ICP is simply:

$$S_o$$

The average service time with ICP is:

$$PS_n + (1 - P)S_o + R_n$$

We are interested in the case when:

$$PS_n + (1 - P)S_o + R_n < S_o$$

which reduces to:

$$P > \frac{R_n}{S_o - S_n}$$

In other words, when the ICP hit ratio is larger than the ratio of the ICP service time to the difference in the HTTP service times, then the benefits of the neighbor hits are greater than the costs of doing the ICP queries. If the ICP hit ratio is smaller than the service time ratio, using ICP does not reduce the overall service time for user requests.

Also note that some people use caches primarily for saving bandwidth, not for reducing service times. They probably don't mind the slight extra delay if it reduces their monthly bandwidth costs.

8.1.3.2 Bandwidth

Speaking of bandwidth, you may be wondering how much of it ICP consumes. It is quite easy to calculate the additional bandwidth ICP uses. The size of an ICP query is 24 bytes plus the length of the URL. The average URL length seems to be about 55 characters, so ICP queries average about 80 bytes. ICP replies are four bytes smaller than queries.

Taken together, an ICP query/reply transaction uses about 160 bytes per neighbor before including UDP and IP headers. A cache with 3 neighbors uses 480 bytes for each cache miss. To put this in perspective, consider that the average web object size is about 10 KB. Thus, a cache with three ICP neighbors increases its bandwidth consumption by about 5% on average. Note that this corresponds to a mea-

To predict hits in a neighbor, a cache needs to know which objects are stored there. A neighbor's cache contents can be represented as a database or directory. These directories must be exchanged between neighbors. By looking up objects in the directory, we can predict cache hits. Since a cache's content changes over time, the directories must also be updated. A primary goal of our new protocol is to make the transmission and storage of this directory as efficient as possible.

Probably the worst we could do is use a plain list of cached URLs. The average URL length is about 55 bytes, or 440 bits. The URL list is poor choice for a directory because URLs are an inefficient representation—they use only printable characters, and some sequences appear very frequently. In information theory terms, URLs have a low amount of entropy. A better option is to use a list of hash values of the URLs. With hash values, we may have collisions where two or more URLs have the same hash value. A collision's probability is related to the hash value range and the number of URLs in the list. For a cache of about one million objects, we probably need 24 to 32 bits per object. That's a significant reduction in directory size compared with 440 bits per object in a URL list. Believe it or not, we can do even better. With an algorithm known as a *Bloom filter*, storage is reduced to only a few bits per object.

8.4.1 Bloom Filters

The Bloom filter algorithm, first described by Burton Bloom in 1970, encodes a set of input keys with the help of some hash functions. Recall that a hash function is simply an algorithm that takes a chunk of data as input (the *key*) and produces an integer as output. A good hash function has the characteristic of being random. That is, given a sufficiently large number of input keys, the output values should have an apparently random distribution. Another hash function characteristic is the range of its output. Often, the range is expressed as a power of 2, such as 32 bits or 2^{32}. One of the most often used hash functions is called *MD5*, which is short for Message Digest (Version 5) [Rivest, 1992]. The MD5 hash function outputs 128-bit values and is used extensively in the current Cache Digest implementation.

A Bloom filter is defined by two parameters, K independent hash functions and an array of M bits. The bits of the filter are used to efficiently encode a collection of N items. For a Cache Digest, the collection is the set of object keys (URIs or URLs) stored in a cache. To construct a Bloom filter, we iterate over the items in the collection. Applying the K hash functions to each item gives us K hash values. Each hash value represents a bit position in the filter that should be turned on. If a hash