A Methodology to Derive SPDY's Initial Dictionary for Zlib Compression

Fan Yang[1], Paul Amer[1], Jonathan Leighton[1], and Mike Belshe[2]

[1]Protocol Engineering Lab, Computer and Information Science Department,
University of Delaware, Newark, DE, USA 19716
{yang, amer, leighton}@cis.udel.edu
[2]Twist, Inc. 650 Mission St., San Francisco, CA, USA 94105
mbelshe@chromium.org

Abstract. This paper presents a methodology to derive an appropriate initial dictionary for SPDY to compress HTTP headers using the dictionary data compression method zlib. After applying our methodology, and by using our proposed initial dictionary, an additional 8% compression of the first HTTP request header and an additional 15% compression of first HTTP reply header on a SPDY connection is gained over SPDY's current default initial dictionary. For the 2nd, 3rd, and further HTTP request (or reply) headers transmitted over the same SPDY connection, compression using our proposed initial dictionary is practically identical to that achieved by SPDY's default initial dictionary. This result suggests zlib's adaptive dictionary evolves to roughly the same state after compressing the first HTTP header regardless of the initial dictionary.

Keywords: adaptive dictionary, compression, HTTP header, SPDY, zlib

1 Introduction

Google is proposing a new application-layer protocol SPDY (pronounced "SPeeDY") as a way of making browsing the Internet faster [1]. HTTP is the application level protocol providing basic request/reply semantics. Unfortunately, HTTP was not designed to minimize latency. One particular HTTP feature, the use of uncompressed request and reply headers, inhibits optimal performance.

SPDY compresses HTTP request and reply headers using zlib [3], a widely-used dictionary-based compression method. In this paper we propose a methodology to derive an appropriate initial zlib dictionary for SPDY to compress HTTP headers.

This paper is organized as follows. Section 2 describes the dictionary technique in data compression, zlib's data compression library, and how SPDY processes HTTP headers before compressing them. We present several design issues and a proposed methodology to derive SPDY's initial dictionary in Sections 3 and 4, respectively. An experiment comparing our proposed initial dictionary with SPDY's default initial dictionary is presented in Section 5. Section 6 concludes the paper.

2 Zlib and SPDY Processing of HTTP Headers

2.1 Dictionary Technique in Data Compression

In many applications, the output of the source consists of recurring patterns. A well understood approach to encoding such sources is to maintain a list, or dictionary, of frequently occurring patterns. When recurring patterns appear in the source output, they are encoded (i.e., compressed) with a reference to the dictionary. An adaptive dictionary method starts with a default initial dictionary and changes its contents during the encoding process, based on the data that is being encoded [2].

2.2 Zlib and Deflate

Zlib is a lossless data compression library which provides in-memory compression and decompression functions [3]. The current zlib library defines methods "deflate" and "inflate" which provide compression and decompression, respectively. These methods can be applied with or without an initial dictionary. The compressed data format, called zlib format, is portable across platforms. Zlib is a crucial component of many operating systems (e.g., Linux, iOS) and gaming consoles (e.g., PlayStation 3).

2.3 Processing HTTP Headers in SPDY

Prior to performing compression, SPDY preprocesses HTTP headers to name/value header blocks (which are part of SPDY control frames). The processing includes the following steps [1]:

1. **Strip Certain Headers.** The Connection, Keep-Alive, and Proxy-Connection headers are not valid and MUST not be sent. Transfer-encoding is no longer valid, because data is already framed at the SPDY layer. For example, in Fig. 1, Keep-Alive and Connection are striped.
2. **Change HTTP Header Field Names to Only Use Lowercase.** By only using lowercase, more redundancy and therefore greater compression are expected. For example, in Fig. 1, the field names "Host", "User-Agent", "Accept", "Accept-Language", "Accept-Encoding" and "Accept-Charset" are changed to lowercase.
3. **Strip Colons and Newlines.** The format of each line other than the first line in an HTTP header is "name: value". For the first line of HTTP request, for example, in Fig. 1, "GET /rsrc.php/v1/y0/r/Se4RZ9IHLuW.css HTTP/1.1" is converted to 3 name/value pairs, i.e., "method: get", "url: /rsrc.php/v1/y0/r/Se4RZ9IHLuW.css" and "version: HTTP/1.1". For the first line of HTTP reply, for example, "HTTP/1.1 200 OK", the name and value pairs are "version: HTTP/1.1" and "status: 200 OK". SPDY only keeps the name/value pairs and strips colons and newlines.
4. **Encode the Strings** (i.e., names and values) **with a 32-bit Integer Length-Prefix**. For example, in Fig. 1, for name "accept-charset", the value 14 is put into "Length of name" field. For value "ISO-8859-1,utf-8;q=0.7,*;q=0.7", the value 30 is put into "Length of value" field. A current proposal to SPDY is to reduce the length-prefix of name to a 16-bit integer.
5. **Put the Number of Name/Value Pairs before All Pairs**. For example, in Fig. 1, there are 9 name/value pairs, so a 32-bit integer 9 is put into "Number of Name/Value pairs" field.

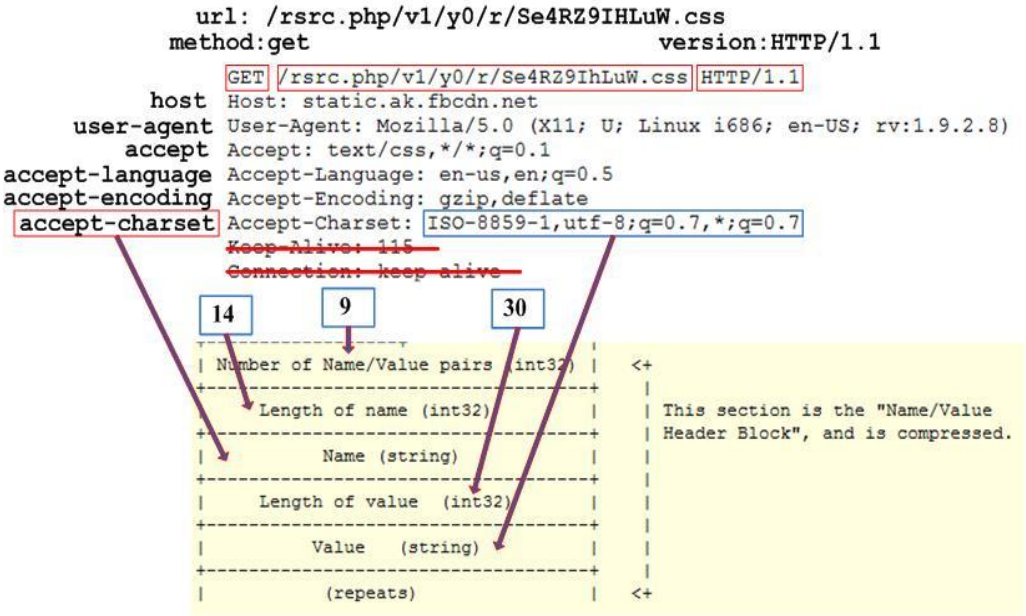

Fig. 1. Processing an HTTP header to name/value header block

After the original HTTP header is converted to a name/value header block, the entire contents of the block is compressed using zlib's deflate. A single zlib stream (context) for all name/value header blocks is maintained in each direction on a connection. In other words, a separate initial dictionary exists and adapts for each direction on a SPDY connection. Each of the two dictionaries is used for the name/value header blocks of all SPDY traffic in one direction.

3 Initial Dictionary Design Issues

Several design issues were considered in developing a methodology to derive an initial dictionary.

3.1 Size of the Initial Dictionary

The current implementation of zlib's deflate method will use at most the window size (number of bytes that can be compressed at the same time) minus 262 bytes of the provided dictionary [3]. The number of bits of window size in SPDY is 11, thus the size of initial dictionary should not exceed 1786 bytes (i.e., 2^{11} - 262). This size limitation prevents including all HTTP specification keywords in the initial

dictionary. Some specified HTTP keywords are rarely used in practice. Thus in our methodology, only keywords regularly used in practice should be included. If and when HTTP keywords change in popularity in the future, a revised initial dictionary can be derived at that time.

3.2 Popular Websites/Browsers Not Included

A conscious decision was made not to include specific popular websites (e.g., facebook, yahoo, Google) or browsers (e.g., firefox), since we do not want to bias the SPDY protocol against new companies/software trying to enter the future marketplace.

3.3 Frequency to Update the Initial Dictionary

An update of zlib's initial dictionary for SPDY requires modifying both peer end points (client browser and server), so changes are problematic. If the initial dictionary is not permitted to change or can only change infrequently (e.g., every few years), we want today's initial dictionary to be appropriate in the future. An obvious example is that the year frequently appears in HTTP headers, so 2011 appears frequently in today's HTTP headers. In two years, however, 2013 will appear frequently and perhaps 2011 not at all. To allow for our initial dictionary to be applicable for several years, we exclude time-dependant keywords (e.g., 2011) from the initial dictionary.

4 Methodology to Derive the Initial Dictionary

The methodology used to derive SPDY's initial dictionary for zlib compression was as follows:

1. Collect HTTP reply headers (in ASCII form) from the main page of the top 1000 websites based on [4] as our training set of HTTP replies. Our training set of HTTP requests was provided by Mike Belshe. These two training sets can be found in [5], [6], respectively.
2. Use punctuation (i.e., blank, comma, newline, semicolon) to parse the headers in both training sets into keywords.
3. Calculate a weight for each keyword in HTTP reply headers by considering the frequency of page views for the top 1000 websites (6th column in [4]). For example, suppose we have three HTTP reply headers (with HTTP version "HTTP/1.1") which are replies from the main pages of facebook.com, youtube.com and yahoo.com, respectively. Then the weight of the keyword "HTTP/1.1" is sum of the frequency of page views of these three websites. Calculate the count of each keyword in HTTP request headers based on the number of appearances of the keyword. For example, suppose we have three HTTP request headers (with method "GET"), then the count of keyword "GET" is three.
4. Build an initial dictionary for HTTP replies based on the weight of each keyword in HTTP reply headers. Build an initial dictionary for HTTP requests based on the count of each keyword in HTTP request headers. Since the SPDY designers preferred to have just a single dictionary for both headers and replies, we then concatenate these two dictionaries. For HTTP header field names and some fixed length values (e.g., HTTP/1.1, 200 OK), we include the 32-bit length prefix before the word. (Note: a separate initial dictionary optimized for either HTTP requests or replies would likely provide better compression, but would in turn make SPDY's implementation more complex.)
5. Add some "known" common non keywords in the dictionary. Example words include HTTP status codes (e.g., 100, 101, 201), months (e.g., Jan, Feb) and days (e.g., Mon, Tue). These words can be called "metadata", which repeat often in HTTP headers and are unlikely to change for a long time.

5 Analysis of Proposed Initial Dictionary

5.1 Experimental Design

To evaluate our proposed initial dictionary, we gathered another set of HTTP request [7] and reply headers [8]. For each of the top 26 websites, we opened the main page (on host *A*) and retrieved the index.html file. Then we collected the first four consecutive HTTP requests for embedded objects (on host *B*) and corresponding four replies, giving a total of 104 request and 104 reply headers. Then, we converted the collected HTTP headers to name/value header blocks. We compressed the name/value header blocks using zlib's deflate with the same parameters (e.g., window size, memory level) as SPDY currently uses. We compared the results of compression using SPDY's current default initial dictionary with that of using our proposed initial dictionary.

Note that the first one of the four consecutive name/value header blocks is compressed using the initial dictionary, while the second one is compressed using the dictionary which has evolved after compressing the first one, and the third one is compressed using the dictionary which has evolved after compressing the first two, etc. A problem existed of how to calculate the compression ratio of each of the four consecutive name/value header blocks independently.

A zlib stream has variables *total_in*, which is the number of (uncompressed) bytes read so far, and *total_out*, which is the number of (compressed) bytes output so far. By example, suppose four consecutive name/value header blocks are 362, 352, 365 and 563 bytes, respectively. We put these four blocks into one file. For the first iteration, we compress the file with *total_in* = 362 and get the *total_out* = 168. Thus the compression ratio of the first block is 168/362. For the second iteration, we compress the file with *total_in* = 714 (size of first two blocks) and get the *total_out* = 238 (the compressed size of first two blocks). Thus the compression ratio of the second block is (238-168)/ 352. For the third iteration, we compress the file with *total_in* = 1079 (sizes of first three blocks) and get the *total_out* = 283 (the compressed size of first three blocks). The compression ratio of the third block is therefore computed as (283-238)/365. The same idea is used to evaluate compression of the fourth header.

5.2 Results

As shown in Table 1, on average for the 104 HTTP request headers, for the first request name/value header block, the compression ratio using SPDY's default initial dictionary is 63.4%, while that of using our proposed initial dictionary is 55.3%, an improvement of roughly 8% more compression. For the 2nd, 3rd and 4th HTTP request name/value header blocks, the difference between using SPDY's default and our proposed initial dictionary is negligible. Both dictionaries improve compression to 10.9% of the original size for the 2nd header, and to less than 6% for the 3rd and 4th headers (and presumably all headers thereafter.) The gain using our proposed initial dictionary is seen only for the first header because after processing the first name/value header block, presumably both the current default and our proposed initial dictionaries evolve to a roughly identical state.

Table 1. Compression results of 104 HTTP request name/value header block

	1st	2nd	3rd	4th
Average original size in bytes	686.2	636.4	631.8	629.8
Average compressed size in bytes using SPDY default initial dictionary (compression ratio)	434.8 (63.4%)	69.3 (10.9%)	36.8 (5.8%)	34.7 (5.5%)
Average compressed size in bytes using our proposed initial dictionary (compression ratio)	379.4 (55.3%)	68.6 (10.9%)	37.0 (5.9%)	35.0 (5.6%)

As shown in Table 2, on average for the 104 HTTP reply headers, for the first reply name/value header block, the compression ratio using SPDY's default dictionary is 60.4%, while that of using our proposed initial dictionary is 45.4%, an improvement of roughly 15% more compression. Similar to the compression of later HTTP request name/value header blocks, for the 2nd, 3rd and 4th HTTP reply name/value header blocks, the difference between using SPDY's default and our proposed initial dictionary is negligible.

Table 2. Compression results of 104 HTTP reply name/value header block

	1st	2nd	3rd	4th
Average original size in bytes	444.6	418.7	431.9	437.7
Average compressed size in bytes using SPDY default initial dictionary (compression ratio)	268.6 (60.4%)	72.5 (17.3%)	54.8 (12.7%)	65.5 (15.0%)
Average compressed size in bytes using proposed initial dictionary (compression ratio)	202.0 (45.4%)	69.2 (16.5%)	54.5 (12.6%)	63.8 (14.6%)

Another perspective considers how many bytes are saved from being transmitted on the Internet. Fig. 2 and Fig. 3 show the results for HTTP request and reply headers respectively. In both figures, the x-axis is the order of name/value header block, and the y-axis is number of bytes each header is

reduced by compression. We can see in Fig. 2, for the first HTTP request name/value header block, roughly 56 more bytes are saved by using our proposed initial dictionary. In Fig. 3, for the first HTTP reply name/value header block, roughly 67 more bytes are saved by using our proposed initial dictionary.

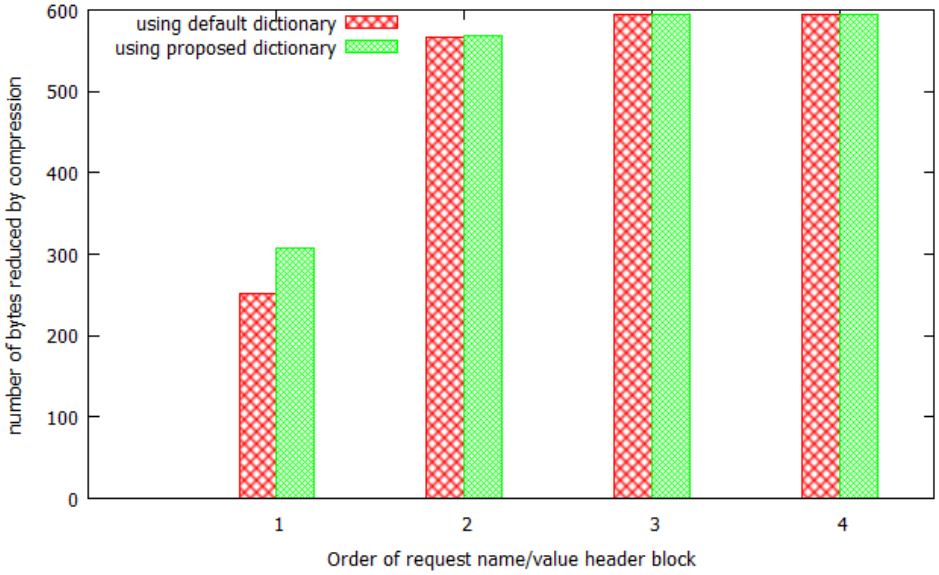

Fig. 2. Average numbers of bytes reduced by compression for request name/value header blocks

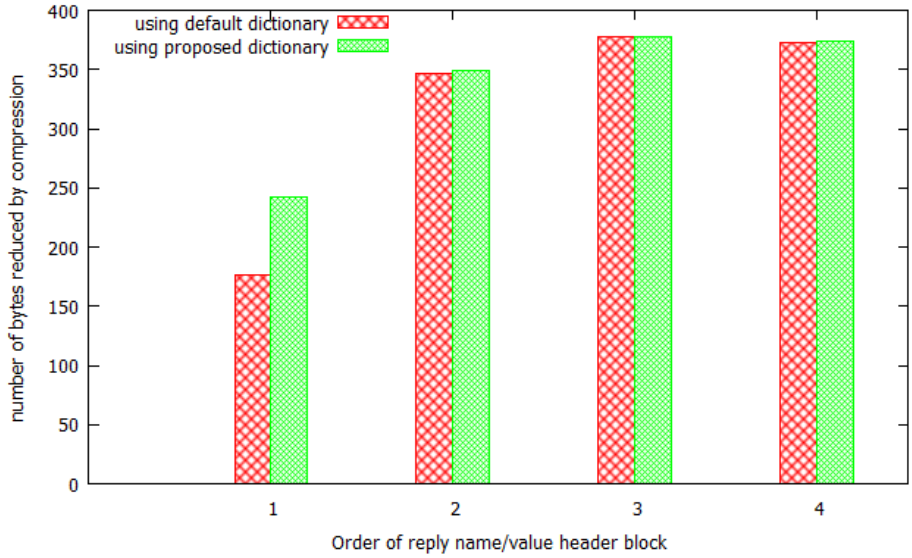

Fig. 3. Average numbers of bytes reduced by compression for reply name/value header blocks

Consider a simple example. Suppose a client opens the main page of website *A* and the main page includes four embedded objects located on website *B*. By using our proposed initial dictionary, roughly 112 more bytes will be saved on average for the requests for *A*'s main page and *B*'s first object. Using our proposed initial dictionary will save roughly 134 bytes when retrieving *A*'s main page and *B*'s first reply.

6 Conclusion

In this research, we designed a methodology to derive an appropriate initial dictionary for SPDY to compress HTTP request and reply headers. The results of experiment show that the compression ratio of the first HTTP request header on a SPDY connection is improved, on average, by 8% over SPDY's current default initial dictionary. Compression of the first reply header is improved, on average, by

15% over SPDY's default initial dictionary. For the 2nd, 3rd and further HTTP request (or reply) headers over the same SPDY connection, the difference between using SPDY's default and our proposed initial dictionary is negligible. This result suggests that zlib's adaptive dictionary evolves to roughly an equivalent state after compressing the first header regardless of the content of the initial dictionary. Our proposed initial dictionary has been accepted and can be found in [1].

References

1 Belshe M, Peon R. SPDY Protocol, http://mbelshe.github.com/SPDY-Specification/draft-mbelshe-spdy-00.xml

2 Sayood K. Introduction to Data Compression (3rd ed.). Morgan Kaufmann; 2005

3 Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library, http://www.zlib.net/

4 The 1000 Most-Visited Sites on the Web by Google, http://www.google.com/adplanner/static/top1000/

5 Training Data: HTTP request headers, http://www.cis.udel.edu/~amer/PEL/SPDY/HTTP_requests_training

6 Training Data: HTTP reply headers, http://www.cis.udel.edu/~amer/PEL/SPDY/HTTP_replies_training

7 Evaluation Data: HTTP request headers, http://www.cis.udel.edu/~amer/PEL/SPDY/HTTP_requests_evaluation

8 Evaluation Data: HTTP reply headers, http://www.cis.udel.edu/~amer/PEL/SPDY/HTTP_replies_evaluation