HTTPbis Working Group
Internet-Draft
Intended status: Informational
Expires: September 19, 2013

R. Peon
Google, Inc

Mar 18, 2013

Header Delta-Compression for HTTP/2.0
draft-rpeon-httpbis-header-compression-03

Abstract

This document describes a mechanism for compressing streams of groups of key-value pairs, often known as Headers in an HTTP session. See RFC 2616 [RFC2616] or successors for more information about headers.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 19, 2013.

Copyright Notice

Table of Contents

1. Overview

There have been several problems pointed out with the use of the gzip compressor in SPDY [SPDY]. The biggest of these problems is that it is possible for a smart attacker to inject content into the compressor, and then to test hypotheses about the prior contents of the compressor by examining the output size after each such content injection. The other issue is that gzip often consumes more CPU than many would like, especially in situations where one is doing forward or reverse proxying. The compressor proposed here intends to solve the first issue and significantly mitigate the second, while providing compression that is not too much worse than gzip.

2. How it works

The 'delta' compressor works by examining the difference between what it is told to compress and the state that it has stored about what it knows about the past. The previous state is encoded in two separate pieces: An LRU of key-value pairs which the compressor 'saw' in the past (including a static group of key-value pairs which every compressor is assumed to have seen), and a set of references into the LRU which is called a header-group, which the compressor uses to determine what has changed between the current input and the past input.

It then encodes this difference by changing the header-group by adding references to stored key-values, and it removes references to key-values which should no longer be part of the output. If a key-value exists in the to-be-compressed data, but is not present in the LRU, then the LRU is modified by having new data added. When new data is added, a reference to that new data is added to the header-group. The mechanism of adding new data takes two forms: Adding an entire new key-value, or by referring to the key part of a stored key-value, and providing a new value.

When the LRU has reached its size limit, The oldest elements are popped off the end, and, any reference to that element is removed.

All keys are assumed to have been lowercased, and if not, will be.

3. Definitions

user-agent: The program or device which a human interacts with directly and which typically initiates the transport layer connection or session

client: Synonym for user-agent

server: The computer or device which typically accepts a connection, stores, and serves data

proxy: An entity acting as a server for the client, and a client for the server

header: A complete set of key-value pairs, either request-headers, or response-headers as defined in RFC2616 [RFC2616] section 5.3 or 6.2, respectively

4. Header pre-processing

4.1. Mapping the first-line

Before the data is input into the compressor (which works only on key-value pairs), the first line of the HTTP message must be made into key-value pairs.

Requests are mapped as follows:
"METHOD PATH VERSION" becomes:

```
[
  (":method",  "METHOD"),
  (":path",    "PATH"),
  (":version", "VERSION")
]
```

Responses are mapped as follows:
"VERSION STATUS-CODE PHRASE" becomes:

```
[
  (":version",     "VERSION"),
  (":status",      "STATUS-CODE"),
  (":status-text", "PHRASE")
]
```

4.2. Mapping HTTP key-values

The rest of the HTTP key-values are simply added to the key-values as mapped from the first-line, with the keys made to be all lowercase, and with cookies split into crumbs by breaking apart the cookie string on semicolons and treating each as if it were a separate header-line.

As an example, the following key-value pairs:

```
                Host: www.foo.com
                User-Agent: Browser/1.x (FooOS; Bar) baz
                Accept-Language: en-US,en;q=0.5
                Cookie: foo;bar; baz
```

become:

```
                [
                  ("host", "www.foo.com")
                  ("user-agent", "Browser/1.x (FooOS; Bar) baz"),
                  ("accept-language", "en-US,en;q=0.5"),
                  ("cookie", "foo"),
                  ("cookie", "bar"),
                  ("cookie", "baz")
                ]
```

5. Compressor and Decompressor State

The header delta de/compression scheme consists of a state machine which executes opcodes, emits output, and modifies internal, persistent, state. The de/compressor state consists of:

static_entries:
: a number of static (unchanging) entries consisting of key-value pairs. These are listed in appendix A.
: e.g. static_entries=[("key1", "val1"), ("key2", "val2), ...]
: an lru-idx references into the static key-value pairs if the value of the lru-idx is < len(static-entries)
: static_entries[lru_idx]

lru:
: a queue of key-value pairs
: e.g. lru = deque([(RefCntString("key1"), "val1"), (RefCntstring("key2", "val2")), ...])
: an lru-idx references a value in the lru if the lru-idx is >= len(static_entries). The mapping of an lru-idx to a offset from the front of the LRU is as follows:

```
            if lru.first_idx > lru_idx:
              queue_idx = 2**16 - lru.first_idx + lru_idx -
len(static_entries)
            else:
              queue_idx = lru_idx - lru.first_idx
```

: The oldest elements of the LRU are popped before inserting a new value if either:

Adding a new entry would exceed the maximum allowable length

Adding a new entry would exceed the maximum allowable byte length

header_groups: (default-size: 1)(max: 255)
: a map of group-id to set of lru-idx. An lru-idx is a reference into either the static key-value pairs or the lru's key-value pairs. The maximum number of header-groups is limited by default to 1, unless a higher-level of the protocol changes this. The maximum number of header_groups is 255. It is not currently allowed to assert that there are '0' allowed header groups.
e.g. header_groups = {0: set([1,4,6,15122]), 1: set([6,76,3], ...)}

lru.first_idx:
: an int indicating the lru-idx of the oldest element in the queue of key-value pairs

max_byte_size: (default: 4k) (max:2**32-1)
: an int indicating the maximum allowable amount of storage used by the strings of the queue's key-value pairs. Unless a higher-level of the protocol changes this, this is assumed to be 4k

max_lru_entries: (default: 1024) (max:2**16-1)
: an int indicating the maximum allowable number of key-value pairs in the queue. Unless a higher-level of the protocol changes this, this is assumed to be 1024

lru.length:
: an int indicating the total number of key-value pairs currently stored in the lru

lru.stored_byte_size:
: an int indicating the total number of bytes of storage used by strings in the queue. Note that the bytes in a ref-counted string are counted only once, regardless of how many times that string is referenced.

6. Header Block Wire Format

The decompressor is fed a header-block which may span multiple HEADERs frames by the HTTP/2 framing layer, the format of which follows:

```
All ints are in network byte order.

header-block: group-id
  ( (ekvsto-opcode ekvsto-count ekvsto-field{ekvsto-count})* |
    (eclone-opcode eclone-count eclone-field{eclone-count})* |
    (etrang-opcode etrang-count etrang-field{etrang-count})* |
    (strang-opcode etrang-count strang-field{etrang-count})* |
    (etoggl-opcode etoggl-count etoggl-field{etoggl-count})* |
    (stoggl-opcode stoggl-count stoggl-field{stoggl-count})* )*
  (clone-opcode clone-count clone-field{clone-count})*
  (kvsto-opcode kvsto-count kvsto-field{kvsto-count})*
  ;

group-id: UINT8;
etoggl-count: UINT8;
stoggl-count: UINT8;
etrang-count: UINT8;
strang-count: UINT8;
eclone-count: UINT8;
sclone-count: UINT8;
ekvsto-count: UINT8;
skvsto-count: UINT8;

stoggl-field: lru-idx;
etoggl-field: lru-idx;
strang-field: lru-idx lru-idx;
etrang-field: lru-idx lru-idx;
eclone-field: lru-idx string;
sclone-field: lru-idx string;
ekvsto-field: string  string;
skvsto-field: string  string;

stoggl-opcode: UINT8(0x00);
etoggl-opcode: UINT8(0x01);
strang-opcode: UINT8(0x02);
etrang-opcode: UINT8(0x03);
skvsto-opcode: UINT8(0x04);
ekvsto-opcode: UINT8(0x05);
sclone-opcode: UINT8(0x06);
eclone-opcode: UINT8(0x07);

lru_idx: UINT16;

string: (HUFFMAN-ENCODED-CHAR)* HUFFMAN-EOF
        padding-to-nearest-byte-boundary;
padding-to-nearest-byte-boundary: 0{0-7};
```

7. String Encoding

Strings are huffman encoded [HUFF] using a canonical huffman coding [CANON]. In the future, the opcode byte will be permuted to allow alternate encodings, such as raw text, binary, or perhaps other options.

The huffman code is constructed by taking the frequency-tables in Appendix B, adding 1 to all entries, then generating a canonical huffman coding. If/while this results in a code with a max-length of greater than 32 bits, divide all frequencies by two, capping the minimum frequency at '1', and regenerate until the max code-length is 32 bits or less. The EOF symbol, when decoded, is represented as 256, which allows for any 8-bit value to be encoded and decoded.

8. Operations

For all operations below, the 's' prefix stands for 'State modifying', whereas the 'e' prefix stands for 'Ephemeral', and does not modify state.

The *kvsto family of opcodes encode a new key-value entirely by providing a new string for key and a new string for val.

The *clone family of opcodes encode a backreference to the key part of a pre-existing key-value from either the static-entries or the lru, and a new string value.

The *toggl family of opcodes encode a backreference to an entire key-value from either the static-entries, or the lru.

The *trang family of opcodes is the same as the toggle family, except that it encodes a range of indices instead of a single index

With four families of opcodes, and two variations (ephemeral vs state-changing) per family, we have eight valid opcodes:

skvsto: (Stateful Key-Value STOre)
: state-modifying kvsto. The new key and value are inserted into the headers and also inserted into the LRU.

ekvsto: (Ephemeral Key-Value STOre)
: ephemeral, non-state-modifying kvsto. The new key and value are inserted into the headers but the LRU is untouched.

sclone (Stateful key CLONE):
state-modifying clone. The key part of the referenced key-value is paired with the new value and inserted into the headers and also inserted into the LRU

eclone (Ephemeral key CLONE):
ephemeral, non-state-modifying clone. The key part of the referenced key-value is paired with the new value and inserted into the headers. No persistent state is modified.

stoggl (Stateful TOGGLe):
state-modifying toggle. If the index exists in the current header group, it will be turned off, else it will be turned on.

etoggl (Ephemeral TOGGLe):
ephemeral, non-state-modifying toggle. If the provided index does not exist in the current header group after all stoggles have modified it, then the key-value as referenced by the provided index will be present in the output, else, that index of the current header group will be temporarily suppressed and will not be included in the headers

strang (Stateful Toggle RANGe):
encodes a range of stoggls

etrang (Ephemeral ToggleRANGe):
encodes a range of etoggls

9. Decompressor algorithm

The pseudo-code below provides a definition of how the header-block is executed by the decompressor.

```
ParseAndExecuteHeaderBlock(header_block):
  store_later = deque()
  etoggles = set()
  stoggles = set()
  headers = dict()
  # the HTTP/2 framing layer determines when the header_block
  # has finished reading.
  group_id  = header_block.read_uint8()
  current_header_group = header_groups[group_id]

  while data in header_block:
    opcode = header_block.read_uint8()
    num_fields = header_block.read_uint8()
    if opcode == stoggl:
```

```
            repeat num_fields times:
              lru_idx = header_block.read_uint16()
              stoggles = set_symmetric_difference(stoggles, [lru_idx])
          elif opcode == etoggl:
            repeat num_fields times:
              lru_idx = header_block.read_uint16()
              etoggles = set_symmetric_difference(etoggles, [lru_idx])
          elif opcode == strang:
            repeat num_fields times:
              lru_idx_first = header_block.read_uint16()
              lru_idx_last = header_block.read_uint16()
              for lru_idx in (lru_idx_first, lru_idx_last) inclusive:
                stoggles = set_symmetric_difference(stoggles, [lru_idx])
                stoggles.add(lru_idx)
          elif opcode == etrang:
            repeat num_fields times:
              lru_idx_first = header_block.read_uint16()
              lru_idx_last = header_block.read_uint16()
              for lru_idx in (lru_idx_first, lru_idx_last) inclusive:
                etoggles = set_symmetric_difference(etoggles, [lru_idx])
          elif opcode == sclone:
            repeat num_fields times:
              lru_idx = header_block.read_uint16()
              val = header_block.read_huffman_string()
              kv = lookup_idx_from_static_entries_or_lru(lru_idx)
              AddToCurrentHeaders(headers, kv.key, val)
              store_later.append(KV(kv.key, val))
          elif opcode == eclone:
            repeat num_fields times:
              lru_idx = header_block.read_uint16()
              val = header_block.read_huffman_string()
              kv = lookup_idx_from_static_entries_or_lru(lru_idx)
              AddToCurrentHeaders(headers, kv.key, val)
          elif opcode == skvsto:
            repeat num_fields times:
              key = header_block.read_huffman_string()
              val = header_block.read_huffman_string()
              AddToCurrentHeaders(headers, key, val)
              store_later.append(KV(key, val))
          elif opcode == ekvsto:
            repeat num_fields times:
              key = header_block.read_huffman_string()
              val = header_block.read_huffman_string()
              AddToCurrentHeaders(headers, key, val)

        # store the state changes to the header-group.
        current_header_group = \
          set_symmetric_difference(current_header_group, stoggles)
```

```
    kv_references = set_symmetric_difference(current_header_group,
                                             etoggles)

    for lru_idx in sorted(kv_references):
      kv = lookup_idx_from_static_entries_or_lru(lru_idx)
      AddToCurrentHeaders(headers, kv.key, kv.val)

    if 'cookie' in headers:
      headers['cookie'] = headers['cookie'].replace('\0', '; ')

    older_headers = []
    for lru_idx in sorted(current_header_group):
      # sorting by idx is suboptimal when the idxs wrap 2**16.
      # As a refinement, we probably want to change this in the
      # future to something which sorts based on the order in
      # which the elements were first mentioned, which can be
      # done by a smart implementation without actually sorting.
      kv = lookup_idx_from_static_entries_or_lru(lru_idx)
      older_headers.append(kv)
    store_later = older_headers + store_later

    # make state changes to the LRU. Note that this may remove
    # items from the header-group if elements that the header-group
    # refers to are removed from the LRU
    for kv in store_later:
      new_lru_idx = lru.store(kv.key, kv.val)

    return headers

  lru.clear():
    while length > 0:
      pop_oldest()

  lru.store(key, val):
    reserve_size = val.size + key.size
    if max_lru_entries == 0 or
       max_byte_size < reserve_size):
       lru.clear()
       return -1
    while length + 1 >= max_lru_entries:
      pop_oldest()
    while true:
      reserve_size = val.size
      if key.refcnt == 1:
        reserve_size += key.size
      if stored_byte_size + reserve_size < max_byte_size:
        break
      pop_oldest()
```

```
    push(KV(key, val))
    new_lru_idx = lru.first + length
    if new_lru_idx >= 2**16:
      new_lru_idx -= 2**16
      new_lru_idx += static_entries.size:
    return new_lru_idx

  lru.pop_oldest():
    kv = queue.front()
    length -= 1
    if kv.key.refcnt == 1:
      stored_byte_size -= kv.key.size
    stored_byte_size -= kv.val.size
    for header_group in header_groups:
      if first_idx in header_group:
        header_group.remove(first_idx)
    first_idx = get_next_idx(first_idx)
    queue.pop_front()

  lru.push(kv):
    length += 1
    if kv.key.refcnt == 1:
      stored_byte_size += kv.key.size
    stored_byte_size += kv.val.size
    queue.push_back(kv)

  lru.get_next_idx(idx):
    idx += 1
    if idx >= 2**16 - 1:
      return decompressor.static_entries.size
    return idx

  lookup_idx_from_static_entries_or_lru(lru_idx):
    if lru_idx < static_entries.size:
      return static_entries[lru_idx]:
    if lru.first_idx > lru_idx:
      queue_idx = 2**16 - lru.first_idx + lru_idx - static_entries.size
    else:
      queue_idx = lru_idx - lru.first_idx
    return lru.queue[queue_idx]
```

10. Compression

The compressor generates a sequence of instructions which the decompressor executes. There are various ways by which the

compressor can determine how to construct these operations. Pseudo-code follows showing one approach. group_id is defined by the sender, but must always be less than the maximum allowed number. A reasonable implementation might assign the same group_id to a set of headers which are likely to be similar, for instance those which go to the same hostname or the same hostname suffix.

```
# assumptions: headers is a dict(), where multiple key:values with
# the same key are encoded as key:value1\0value2\0value3...
# group_id is provided by some other implementation-dependent
# code

# This compressor does not use all of the opcodes and serves simply
# as an example of a workable, if suboptimal, implementation
MakeOperations(self, headers, group_id):
  headers_set = set()
  for (key, val) in headers:
    splittoken = '\0'
    if key == 'cookie':
      splittoken = ';'
    for partial_val in split(val, ';'):
      headers_set.add( (key, partial_val) )
      # Note that this discards duplicates.
      # If we decide we care about that generate an 'ekvsto' or
      # 'eclone' for that (duplicate) key-value here.

  keep_set = set()
  done_set = set()
  for idx in header_groups[group_id]:
    kv = lookup_idx_from_static_entries_or_lru(idx)
    if kv in headers_set:
      # If the KV referenced by the idx in the header-group
      # is also in the to-be-compressed headers, then we
      # keep using that reference (don't remove it from the
      # header-group)
      keep_set.add(idx)  # we want to keep this one
      headers_set.remove(kv)
    else:
      # If we're not finding the KV referenced by the idx in
      # the header-group in the to-be-compressed-header, then
      # this idx needs to be removed from the header-group.
      done_set.add(idx)  # we'll want to remove it

  instructions = dict()
  toggls = set()
  clones = []
  kvstos = []
  erefs = []
```

```
    for (key, val) in headers_set:
      # The following 'if' block is a demonstration of an
      # optimization-- since path and referer are rarely
      # backreferenced, and since they are often large and
      # they would, if included in the LRU, cause other entries
      # to be expired from the LRU, we ensure that these don't
      # get stored in the LRU by emitting an 'ephemeral' operation
      if key in [":path", "referer"]:
        instructions['ekvsto'].append( (key, val) )
        continue
      # FindEntryIdx looks for a matching key-value in the LRU, and then
      # in the static-entries, recording the first matching key it finds
      # while searching for the whole match. If it does find a whole
      # match then v_idx will be valid. If it finds an entry with a key
      # which matches, then k_idx will be valid.
      (k_idx, v_idx) = FindEntryIdx(key, val)
      if both k_idx and v_idx are valid:
        # if we found a index for all of the kv, we'll generate
        # a new toggle which backreferences that entire kv.
        toggls.add(v_idx)
      elif only k_idx is valid:
        # Otherwise, if we did't find all of the kv pre-existing,
        # but there was something that already had that key,
        # generate a clone, which backreferences the key and provides
        # a new value.
        instructions['sclone'].append( (k_idx, val) )
      else:
        # Otherwise, we'll need to store a new key and value, both.
        instructions['skvstos'].append( (key, val) )

    full_toggl_list = union(toggls, done_set)
    # convert runs of toggls into trangs
    (trangs, toggls) = ComputeTrangsFromRawToggles(full_toggl_list)
    instructions['stoggl'] = toggls
    instructions['strang'] = trangs

    header_block = SerializeInstructions(instructions, group_id)

    # Execute the instructions just like you would when decompressing.
    # We're throwing away the computed headers here, because all
    # we care about is the side-effects to the header_groups and the
    # lru from executing the generated instructions.
    ParseAndExecuteHeaderBlock(header_block)
  return header_block


SerializeInstructions(instructions, group_id):
  outbuf.write_uint8(group_id)
```

```
        for opcode in ['stoggl', 'etoggl',
                       'strang', 'etrang',
                       'eclone', 'ekvsto',
                       'sclone', 'skvsto']:
          if not opcode in instructions:
            continue
          ops_idx = 0
          ops_len = len(instructions[opcode])
          while ops_len > ops_idx:
            ops_to_go = ops_len - ops_idx
            outbuf.write_uint8(OpcodeToOpcodeVal(opcode))
            # a value of '0' in this field means '1'.
            # a value of '255' in this field means '256',
            # thus, subtract one from the actual value when
            # preparing to write to the wire.
            outbuf.write_uint8(min(256, ops_to_go) - 1)
            orig_idx = ops_idx
            for i in xrange(ops_to_go):
              if opcode in ['stoggl', 'etoggl']:
                outbuf.write_uint16(instructions[ops_idx])
              elif opcode in ['strang', 'etrang']:
                outbuf.write_uint16(instructions[ops_idx][0])
                outbuf.write_uint16(instructions[ops_idx][1])
              elif opcide in ['sclone', 'eclone']:
                outbuf.write_uint16(instructions[ops_idx][0])
                outbuf.encode_and_write_string(instructions[ops_idx][1])
              elif opcode in ['skvsto', 'ekvsto']:
                outbuf.encode_and_write_string(instructions[ops_idx][0])
                outbuf.encode_and_write_string(instructions[ops_idx][1])
              ops_idx += 1
        return outbuf
```

If the resulting output buffer is larger than the maximum allowed frame size, then the buffer shall be split into maximum-allowed-payload-size or smaller sections, and sent in separate HEADERS frames, with only the last indicating that the frame is finished by asserting the FRAME_FINISHED flag.

11. Example

11.1. Background

Here is a simple example showing an input, the changing part of the compressor state, and an ascii-ified version of what would be serialized on the wire.

```
GET / HTTP/1.0
Host: www.foo.com
User-Agent: bar-ua baz stuff
Accept-Language: en-US,en;q=0.5
```

The first stage of processing is to break the first line into key-value pairs. The first line becomes:

```
[
(":method": "GET"),
(":path": "/"),
(":version": "HTTP/1.0")
]
```

This gets integrated with the rest of the headers, becoming:

```
[
(":method", "GET"),
(":path", "/"),
(":version", "HTTP/1.0"),
("host", "www.foo.com"),
("user-agent", "bar-ua baz stuff"),
("accept-language", "en-US,en;q=0.5"),
]
```

The compressor now goes through each key-value, determining if it is already present in the header-group, in the LRU or static state, and determines what it needs to emit.

11.2. Example Serialization

This is sample output from a program which implements the compression specification above. It prints out the stream ID, then the group ID, then the instructions that the encoder created, then the serialized form of these instructions, followed last by the decompressed output.

```
* http2-demo: 2 req messages
##############################################################################
# delta2 request 1 (of 2) for http2-demo

stream_id: 1234 group_id: 1
{'opcode': 'eclone', 'index':     0, 'val': '/http2_sample.html'}
{'opcode': 'stoggl', 'index':     1}
{'opcode': 'stoggl', 'index':     3}
{'opcode': 'sclone', 'index':    38, 'val': 'no-cache'}
{'opcode': 'sclone', 'index':    10, 'val': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3'}
```

```
{'opcode': 'sclone', 'index':    11, 'val': 'gzip,deflate,sdch'}
{'opcode': 'sclone', 'index':     9, 'val': 'text/html,application/
xhtml+xml,application/xml;q=0.9,*/*;q=0.8'}
{'opcode': 'sclone', 'index':    16, 'val': 'no-cache'}
{'opcode': 'sclone', 'index':     4, 'val': 'http2-demo'}
{'opcode': 'sclone', 'index':    12, 'val': 'en-US,en;q=0.8'}
{'opcode': 'sclone', 'index':    51, 'val': 'Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.28 Safari/537.31'}


00 FE 08 01 80 00 04 D2 01 05 00 00 00 0C C8 9E | ................
9B A2 AD 5F A0 9B 32 D7 49 00 00 01 00 01 00 03 | ..._..2.I.......
04 07 00 26 6B A7 5A 97 98 C8 00 0A F6 FD 3E 33 | ...&k.Z.......>3
F3 E7 4D 73 A3 F8 D8 B1 9F 9F A3 F5 69 CD 57 F1 | ..Ms........i.W.
FF 5E 8F D5 A7 35 12 00 0B CB F6 C7 FF 18 0E 3A | .^...5.........:
28 87 F8 8E 0B CE 40 00 09 21 F2 20 CC B5 D3 F8 | (.....@..!. ....
53 DF A3 16 A2 63 9A 1E 59 96 BA 7F DF 96 BA 7F | S....c..Y.......
0A 7B F4 62 D4 4C 73 43 CB 5D 3D 1F AB 4E 6A FF | .{.b.LsC.]=..Nj.
8F FA 0F FA F4 7E AD 39 B9 C8 00 10 6B A7 5A 97 | .....~.9....k.Z.
98 C8 00 04 CC 89 E9 9F 01 D5 D2 00 0C 16 CF F6 | ................
FA 7F 02 DF 47 EA D3 9B 9C 80 00 33 F7 BB F6 CD | ....G......3....
34 50 52 63 FF B7 FC 7E 50 8F 47 FB 7B 98 DD BC | 4PRc...~P.G.{...
BF DB CB 9F 2B B9 71 FF 9F F6 EC 7B F4 1F C0 DF | ....+.q....{....
FE 98 41 4D 15 1A 84 7F B7 FC 7F AF 67 D7 DF EE | ..AM........g...
FE 3F DB 46 78 0F FB 7A C5 7E 0E FF 9F F6 EE CE | .?.Fx..z.~......
0E D4 41 3C 8C 73 23 8A 0E 64 F3 FF 6F A2 B1 54 | ..A<.s#..d..o..T
18 14 D1 51 A8 44 80                            | ...Q.D.

###### decompressed ######
get /http2_sample.html HTTP/1.1
accept-language: en-US,en;q=0.8
accept-encoding: gzip,deflate,sdch
accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
pragma: no-cache
cache-control: no-cache
host: http2-demo

################################################################################
# delta2 request 2 (of 2) for http2-demo

stream_id: 1234 group_id: 1
{'opcode': 'eclone', 'index':     0, 'val': '/s/http2_fractal.jpg'}
{'opcode': 'sclone', 'index':    42, 'val': 'http://http2-demo/
http2_sample.html'}
```

```
{'opcode': 'sclone', 'index':    70, 'val': '*/*'}
{'opcode': 'strang', 'index':    69, 'index_start':    67}
{'opcode': 'strang', 'index':    74, 'index_start':    71}
```

```
00 3E 08 01 80 00 04 D2 01 05 00 00 00 08 86 64 | .>.............d
4F 4D D8 C1 4B 25 68 6E AF CA 40 04 01 00 2A CC | OM..K%hn..@...*.
89 F6 00 66 44 F4 CF 80 EA E0 CC 89 E9 BA 2A D5 | ...fD.........*.
FA 09 B3 2D 74 90 00 46 FF A0 FF A9 00 02 01 00 | ...-t..F........
45 00 43 00 4A 00 47                            | E.C.J.G

###### decompressed ######
get /s/http2_fractal.jpg HTTP/1.1
accept-language: en-US,en;q=0.8
pragma: no-cache
accept: */*
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
referer: http://http2-demo/http2_sample.html
cache-control: no-cache
host: http2-demo
accept-encoding: gzip,deflate,sdch

                  size  time | ratio min   max   std
  delta2           334  0.00 | 1.00  1.00  1.00  0.00

* http2-demo: 2 res messages
################################################################################
# delta2 response 1 (of 2) for http2-demo

stream_id: 1234 group_id: 1
{'opcode': 'stoggl', 'index':     6}
{'opcode': 'sclone', 'index':    16, 'val': 'public, max-age=3600000000'}
{'opcode': 'sclone', 'index':    27, 'val': 'Fri, 1 Jan 2100 12:00:00 GMT'}
{'opcode': 'sclone', 'index':    18, 'val': 'gzip'}
{'opcode': 'sclone', 'index':    13, 'val': 'bytes'}
{'opcode': 'sclone', 'index':    52, 'val': 'Accept-Encoding'}
{'opcode': 'sclone', 'index':    19, 'val': '797'}
{'opcode': 'sclone', 'index':    34, 'val': 'Thu, 08 Nov 2012 17:24:16 GMT'}
{'opcode': 'sclone', 'index':    24, 'val': 'Tue, 12 Mar 2013 23:12:44 GMT'}
{'opcode': 'sclone', 'index':    44, 'val': 'Apache/2.2.22 (Ubuntu)'}
{'opcode': 'sclone', 'index':    23, 'val': 'text/html'}


00 99 08 01 80 00 04 D2 01 00 00 00 06 04 09 00 | ................
10 C3 3D BC 6D AE 60 E5 0F 39 E1 BE 3A 91 40 88 | ..=.m.`..9..:.@.
88 88 8C 80 00 1B EC C6 D9 80 83 B6 17 01 90 88 | ................
11 B8 45 C2 21 4D 4F 90 00 12 DF FB B7 09 00 00 | ..E.!MO.........
0D DB E9 94 79 C8 00 34 D7 5D 71 C3 29 FA EE AE | ....y..4.]q.)...
FB 2D BB 7C 80 00 13 5B 57 20 00 22 7F 0C E6 01 | .-.|...[W .".....
60 77 5F E2 06 24 60 4B 71 A5 C5 40 53 53 E4 00 | `w_..$`Kq..@SS..
```

18 7E 71 98 08 C2 A8 62 06 24 80 34 38 8D C9 48 | .~q....b.$.48..H

```
53 53 E4 00 2C D7 84 2B E1 1E 83 D2 7A 4C C3 D7 | SS..,..+....zL..
F5 DB 9D D9 67 EC 90 00 17 CA 3E 79 74 70 CB 97 | ....g.....>ytp..
19 00                                           | ..

###### decompressed ######
HTTP/1.1 200 ?
content-length: 797
content-encoding: gzip
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
vary: Accept-Encoding
server: Apache/2.2.22 (Ubuntu)
last-modified: Thu, 08 Nov 2012 17:24:16 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: text/html

################################################################################
# delta2 response 2 (of 2) for http2-demo

stream_id: 1234 group_id: 1
{'opcode': 'stoggl', 'index':    69}
{'opcode': 'sclone', 'index':    75, 'val': 'image/jpeg'}
{'opcode': 'sclone', 'index':    71, 'val': '365'}
{'opcode': 'sclone', 'index':    72, 'val': 'Tue, 23 Oct 2012 02:26:33 GMT'}
{'opcode': 'strang', 'index':    67, 'index_start':    66}
{'opcode': 'strang', 'index':    74, 'index_start':    73}


00 35 08 01 80 00 04 D2 01 00 00 00 45 04 02 00 | .5..........E...
4B B7 94 37 C7 A3 F1 84 77 C8 00 47 45 0A 90 00 | K..7....w..GE...
48 7E 71 98 0D 01 DF 5E 40 62 46 02 6E 3A 1C 84 | H~q....^@bF.n:..
05 35 3E 40 02 01 00 43 00 42 00 4A 00 49       | .5>@...C.B.J.I

###### decompressed ######
HTTP/1.1 200 ?
content-length: 365
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
server: Apache/2.2.22 (Ubuntu)
last-modified: Tue, 23 Oct 2012 02:26:33 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: image/jpeg

                  size  time | ratio min   max   std
  delta2           224  0.00 | 1.00  1.00  1.00  0.00
```

```
* http2-demo1: 5 req messages
################################################################################
# delta2 request 1 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'eclone', 'index':     0, 'val': '/s/0.png'}
{'opcode': 'sclone', 'index':    81, 'val': 'http2-demo1'}
{'opcode': 'strang', 'index':    80, 'index_start':    75}
{'opcode': 'strang', 'index':    85, 'index_start':    82}


00 20 08 01 80 00 04 D2 02 05 00 00 00 08 81 CC | . ..............
F6 E5 20 04 00 00 51 CC 89 E9 9F 01 D5 C8 90 02 | .. ...Q.........
01 00 50 00 4B 00 55 00 52                      | ..P.K.U.R

###### decompressed ######
get /s/0.png HTTP/1.1
accept-language: en-US,en;q=0.8
accept: */*
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
referer: http://http2-demo/http2_sample.html
pragma: no-cache
cache-control: no-cache
host: http2-demo1
accept-encoding: gzip,deflate,sdch

################################################################################
# delta2 request 2 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'eclone', 'index':     0, 'val': '/s/6.png'}
{'opcode': 'stoggl', 'index':    96}


00 0E 08 01 80 00 04 D2 02 05 00 00 00 08 87 23 | ...............#
3D B9 48 00 00 00 60                            | =.H...`

###### decompressed ######
get /s/6.png HTTP/1.1
accept-language: en-US,en;q=0.8
accept: */*
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
referer: http://http2-demo/http2_sample.html
```

```
pragma: no-cache
```

```
cache-control: no-cache
host: http2-demo1
accept-encoding: gzip,deflate,sdch

################################################################################
# delta2 request 3 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'eclone', 'index':     0, 'val': '/s/12.png'}


00 0B 08 01 80 00 04 D2 02 05 00 00 00 08 82 12 | ................
67 B7 29 00                                     | g.).

###### decompressed ######
get /s/12.png HTTP/1.1
accept-language: en-US,en;q=0.8
accept: */*
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
referer: http://http2-demo/http2_sample.html
pragma: no-cache
cache-control: no-cache
host: http2-demo1
accept-encoding: gzip,deflate,sdch

################################################################################
# delta2 request 4 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'eclone', 'index':     0, 'val': '/s/18.png'}


00 0B 08 01 80 00 04 D2 02 05 00 00 00 08 82 39 | ...............9
99 ED CA 40                                     | ...@

###### decompressed ######
get /s/18.png HTTP/1.1
accept-language: en-US,en;q=0.8
accept: */*
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
referer: http://http2-demo/http2_sample.html
pragma: no-cache
cache-control: no-cache
```

```
host: http2-demo1
```

```
accept-encoding: gzip,deflate,sdch

################################################################################
# delta2 request 5 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'eclone', 'index':     0, 'val': '/s/24.png'}


00 0B 08 01 80 00 04 D2 02 05 00 00 00 08 82 78 | ...............x
99 ED CA 40                                     | ...@


###### decompressed ######
get /s/24.png HTTP/1.1
accept-language: en-US,en;q=0.8
accept: */*
user-agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like
Gecko) Chrome/26.0.1410.28 Safari/537.31
:scheme: http
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
referer: http://http2-demo/http2_sample.html
pragma: no-cache
cache-control: no-cache
host: http2-demo1
accept-encoding: gzip,deflate,sdch


* http2-demo1: 5 res messages
################################################################################
# delta2 response 1 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'sclone', 'index':    82, 'val': 'image/png'}
{'opcode': 'sclone', 'index':    84, 'val': 'Sat, 23 Jun 2012 02:03:47 GMT'}
{'opcode': 'sclone', 'index':    83, 'val': '338'}
{'opcode': 'strang', 'index':    81, 'index_start':    76}


00 2C 08 01 80 00 04 D2 02 04 02 00 52 B7 94 37 | .,..........R..7
C7 A3 0B B7 C8 00 54 D9 0C A6 03 40 76 E7 70 18 | ......T....@v.p.
91 80 9B 85 0E 4A C2 9A 9F 20 00 53 42 19 20 02 | .....J... .SB. .
00 00 51 00 4C                                  | ..Q.L

###### decompressed ######
HTTP/1.1 200 ?
content-length: 338
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
```

```
server: Apache/2.2.22 (Ubuntu)
last-modified: Sat, 23 Jun 2012 02:03:47 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: image/png

################################################################################
# delta2 response 2 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'strang', 'index':    93, 'index_start':    91}


00 06 08 01 80 00 04 D2 02 02 00 00 5D 00 5B    | ............].[

###### decompressed ######
HTTP/1.1 200 ?
content-length: 338
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
server: Apache/2.2.22 (Ubuntu)
last-modified: Sat, 23 Jun 2012 02:03:47 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: image/png

################################################################################
# delta2 response 3 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'stoggl', 'index':    93}
{'opcode': 'sclone', 'index':   102, 'val': '358'}


00 0B 08 01 80 00 04 D2 02 00 00 00 5D 04 00 00 | ............]...
66 42 99 20                                     | fB.

###### decompressed ######
HTTP/1.1 200 ?
content-length: 358
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
server: Apache/2.2.22 (Ubuntu)
last-modified: Sat, 23 Jun 2012 02:03:47 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: image/png
```

```
################################################################################
# delta2 response 4 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'sclone', 'index':   111, 'val': '397'}


00 07 08 01 80 00 04 D2 02 04 00 00 6F 43 57 20 | ............oCW

###### decompressed ######
HTTP/1.1 200 ?
content-length: 397
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
server: Apache/2.2.22 (Ubuntu)
last-modified: Sat, 23 Jun 2012 02:03:47 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: image/png

################################################################################
# delta2 response 5 (of 5) for http2-demo1

stream_id: 1234 group_id: 2
{'opcode': 'stoggl', 'index':    92}
{'opcode': 'sclone', 'index':   119, 'val': 'Sat, 23 Jun 2012 02:03:46 GMT'}
{'opcode': 'sclone', 'index':   120, 'val': '345'}


00 20 08 01 80 00 04 D2 02 00 00 00 5C 04 01 00 | . ..........\...
77 D9 0C A6 03 40 76 E7 70 18 91 80 9B 85 0E 4D | w....@v.p......M
01 4D 4F 90 00 78 42 55 20                      | .MO..xBU

###### decompressed ######
HTTP/1.1 200 ?
content-length: 345
accept-ranges: bytes
expires: Fri, 1 Jan 2100 12:00:00 GMT
server: Apache/2.2.22 (Ubuntu)
last-modified: Sat, 23 Jun 2012 02:03:46 GMT
cache-control: public, max-age=3600000000
date: Tue, 12 Mar 2013 23:12:44 GMT
content-type: image/png
```

12. Unfinished components

These are components that must be added in the future.

Encoding of raw or ascii bytes

Describing safe mechanisms for changing the allowable compressor state size downwards.

The frequency table used to generate the huffman encoding should be updated with a more comprehensive analysis of header-character frequency.

13. Security Considerations

The compressor algorithm described here is expected to be immune to the current attacks against encrypted stream-based compressors such as TLS+gzip, but more scrutiny is warranted. The reason that it is believed that the algorithm(s) expressed here is immune is that any backreference to a header key or value always requires a whole-text match, and thus any probe of the compression context confirms no hypothesis unless the attacker has guessed the entire plaintext key and value simultaneously.

14. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

15. Acknowledgements

16. Appendix A

Appendix A (static-entries) :

```
# (key, val)
# Order does matter...
static_entries = [
    (':path', '/'),
    (':scheme', 'http'),
    (':scheme', 'https'),
    (':method', 'get'),
    (':host', ''),
```

```
                ('cookie', ''),
                (':status', '200'),
                (':status-text', 'OK'),
                (':version', '1.1'),
                ('accept', ''),
                ('accept-charset', ''),
                ('accept-encoding', ''),
                ('accept-language', ''),
                ('accept-ranges', ''),
                ('allow', ''),
                ('authorizations', ''),
                ('cache-control', ''),
                ('content-base', ''),
                ('content-encoding', ''),
                ('content-length', ''),
                ('content-location', ''),
                ('content-md5', ''),
                ('content-range', ''),
                ('content-type', ''),
                ('date', ''),
                ('etag', ''),
                ('expect', ''),
                ('expires', ''),
                ('from', ''),
                ('if-match', ''),
                ('if-modified-since', ''),
                ('if-none-match', ''),
                ('if-range', ''),
                ('if-unmodified-since', ''),
                ('last-modified', ''),
                ('location', ''),
                ('max-forwards', ''),
                ('origin', ''),
                ('pragma', ''),
                ('proxy-authenticate', ''),
                ('proxy-authorization', ''),
                ('range', ''),
                ('referer', ''),
                ('retry-after', ''),
                ('server', ''),
                ('set-cookie', ''),
                ('status', ''),
                ('te', ''),
                ('trailer', ''),
                ('transfer-encoding', ''),
                ('upgrade', ''),
                ('user-agent', ''),
                ('vary', ''),
```

```
            ('via', ''),
            ('warning', ''),
            ('www-authenticate', ''),
            ('access-control-allow-origin', ''),
            ('content-disposition', ''),
            ('get-dictionary', ''),
            ('p3p', ''),
            ('x-content-type-options', ''),
            ('x-frame-options', ''),
            ('x-powered-by', ''),
            ('x-xss-protection', ''),
            ]
```

17. Appendix B

Appendix B huffman code-table for requests

sym	aligned to MSB as bits	len	aligned to LSB as hex	len
(0)	|11111111|11111111|11110111|100	[27]	7ffffbc	[27]
(1)	|11111111|11111111|11110111|101	[27]	7ffffbd	[27]
(2)	|11111111|11111111|11110111|110	[27]	7ffffbe	[27]
(3)	|11111111|11111111|11110111|111	[27]	7ffffbf	[27]
(4)	|11111111|11111111|11111000|000	[27]	7ffffc0	[27]
(5)	|11111111|11111111|11111000|001	[27]	7ffffc1	[27]
(6)	|11111111|11111111|11111000|010	[27]	7ffffc2	[27]
(7)	|11111111|11111111|11111000|011	[27]	7ffffc3	[27]
(8)	|11111111|11111111|11111000|100	[27]	7ffffc4	[27]
(9)	|11111111|11111111|11111000|101	[27]	7ffffc5	[27]
(10)	|11111111|11111111|11111000|110	[27]	7ffffc6	[27]
(11)	|11111111|11111111|11111000|111	[27]	7ffffc7	[27]
(12)	|11111111|11111111|11111001|000	[27]	7ffffc8	[27]
(13)	|11111111|11111111|11111001|001	[27]	7ffffc9	[27]
(14)	|11111111|11111111|11111001|010	[27]	7ffffca	[27]
(15)	|11111111|11111111|11111001|011	[27]	7ffffcb	[27]
(16)	|11111111|11111111|11111001|100	[27]	7ffffcc	[27]
(17)	|11111111|11111111|11111001|101	[27]	7ffffcd	[27]
(18)	|11111111|11111111|11111001|110	[27]	7ffffce	[27]
(19)	|11111111|11111111|11111001|111	[27]	7ffffcf	[27]
(20)	|11111111|11111111|11111010|000	[27]	7ffffd0	[27]
(21)	|11111111|11111111|11111010|001	[27]	7ffffd1	[27]
(22)	|11111111|11111111|11111010|010	[27]	7ffffd2	[27]
(23)	|11111111|11111111|11111010|011	[27]	7ffffd3	[27]
(24)	|11111111|11111111|11111010|100	[27]	7ffffd4	[27]

```
    ( 25) |11111111|11111111|11111010|101 [27]      7fffd5 [27]
    ( 26) |11111111|11111111|11111010|110 [27]      7fffd6 [27]
    ( 27) |11111111|11111111|11111010|111 [27]      7fffd7 [27]
    ( 28) |11111111|11111111|11111011|000 [27]      7fffd8 [27]
    ( 29) |11111111|11111111|11111011|001 [27]      7fffd9 [27]
    ( 30) |11111111|11111111|11111011|010 [27]      7fffda [27]
    ( 31) |11111111|11111111|11111011|011 [27]      7fffdb [27]
' ' ( 32) |11111111|0110 [12]                           ff6 [12]
'!' ( 33) |11111111|0111 [12]                           ff7 [12]
'"' ( 34) |11111111|111010 [14]                        3ffa [14]
'#' ( 35) |11111111|1111100 [15]                       7ffc [15]
'$' ( 36) |11111111|1111101 [15]                       7ffd [15]
'%' ( 37) |100110 [6]                                    26 [6]
'&' ( 38) |1110000 [7]                                   70 [7]
''' ( 39) |11111111|1111110 [15]                       7ffe [15]
'(' ( 40) |11111111|1000 [12]                           ff8 [12]
')' ( 41) |11111111|1001 [12]                           ff9 [12]
'*' ( 42) |11111111|1010 [12]                           ffa [12]
'+' ( 43) |11111111|1011 [12]                           ffb [12]
',' ( 44) |11111110|00 [10]                             3f8 [10]
'-' ( 45) |100111 [6]                                    27 [6]
'.' ( 46) |00110 [5]                                      6 [5]
'/' ( 47) |0000 [4]                                       0 [4]
'0' ( 48) |00111 [5]                                      7 [5]
'1' ( 49) |01000 [5]                                      8 [5]
'2' ( 50) |01001 [5]                                      9 [5]
'3' ( 51) |101000 [6]                                    28 [6]
'4' ( 52) |1110001 [7]                                   71 [7]
'5' ( 53) |101001 [6]                                    29 [6]
'6' ( 54) |1110010 [7]                                   72 [7]
'7' ( 55) |101010 [6]                                    2a [6]
'8' ( 56) |1110011 [7]                                   73 [7]
'9' ( 57) |101011 [6]                                    2b [6]
':' ( 58) |101100 [6]                                    2c [6]
';' ( 59) |11110100|0 [9]                               1e8 [9]
'<' ( 60) |11111111|11111111|10 [18]                  3fffe [18]
'=' ( 61) |101101 [6]                                    2d [6]
'>' ( 62) |11111111|11111110|0 [17]                   1fffc [17]
'?' ( 63) |11110100|1 [9]                                1e9 [9]
'@' ( 64) |11111111|11100 [13]                          1ffc [13]
'A' ( 65) |11101100 [8]                                  ec [8]
'B' ( 66) |11101101 [8]                                  ed [8]
'C' ( 67) |11101110 [8]                                  ee [8]
'D' ( 68) |11101111 [8]                                  ef [8]
'E' ( 69) |11110101|0 [9]                               1ea [9]
'F' ( 70) |1110100 [7]                                   74 [7]
'G' ( 71) |11110101|1 [9]                               1eb [9]
'H' ( 72) |11110110|0 [9]                               1ec [9]
```

```
'I' ( 73) |11110110|1 [9]                                  1ed [9]
'J' ( 74) |11111110|01 [10]                                3f9 [10]
'K' ( 75) |11111111|010 [11]                               7fa [11]
'L' ( 76) |11110111|0 [9]                                  1ee [9]
'M' ( 77) |11110111|1 [9]                                  1ef [9]
'N' ( 78) |11111000|0 [9]                                  1f0 [9]
'O' ( 79) |11111000|1 [9]                                  1f1 [9]
'P' ( 80) |11111001|0 [9]                                  1f2 [9]
'Q' ( 81) |11111110|10 [10]                                3fa [10]
'R' ( 82) |11111001|1 [9]                                  1f3 [9]
'S' ( 83) |11111010|0 [9]                                  1f4 [9]
'T' ( 84) |11111010|1 [9]                                  1f5 [9]
'U' ( 85) |11111011|0 [9]                                  1f6 [9]
'V' ( 86) |11111011|1 [9]                                  1f7 [9]
'W' ( 87) |11111100|0 [9]                                  1f8 [9]
'X' ( 88) |11111100|1 [9]                                  1f9 [9]
'Y' ( 89) |11111110|11 [10]                                3fb [10]
'Z' ( 90) |11111111|00 [10]                                3fc [10]
'[' ( 91) |11111111|111011 [14]                           3ffb [14]
'\' ( 92) |11111111|11111111|11111011|100 [27]      7ffffdc [27]
']' ( 93) |11111111|111100 [14]                           3ffc [14]
'^' ( 94) |11111111|111101 [14]                           3ffd [14]
'_' ( 95) |101110 [6]                                      2e [6]
'`' ( 96) |11111111|11111111|110 [19]                   7fffe [19]
'a' ( 97) |01010 [5]                                        a [5]
'b' ( 98) |101111 [6]                                      2f [6]
'c' ( 99) |01011 [5]                                        b [5]
'd' (100) |110000 [6]                                      30 [6]
'e' (101) |0001 [4]                                         1 [4]
'f' (102) |110001 [6]                                      31 [6]
'g' (103) |110010 [6]                                      32 [6]
'h' (104) |110011 [6]                                      33 [6]
'i' (105) |01100 [5]                                        c [5]
'j' (106) |1110101 [7]                                     75 [7]
'k' (107) |11110000 [8]                                    f0 [8]
'l' (108) |110100 [6]                                      34 [6]
'm' (109) |110101 [6]                                      35 [6]
'n' (110) |01101 [5]                                        d [5]
'o' (111) |01110 [5]                                        e [5]
'p' (112) |01111 [5]                                        f [5]
'q' (113) |11111101|0 [9]                                  1fa [9]
'r' (114) |10000 [5]                                       10 [5]
's' (115) |10001 [5]                                       11 [5]
't' (116) |0010 [4]                                         2 [4]
'u' (117) |110110 [6]                                      36 [6]
'v' (118) |11110001 [8]                                    f1 [8]
'w' (119) |110111 [6]                                      37 [6]
'x' (120) |11110010 [8]                                    f2 [8]
```

```
     'y' (121) |11110011 [8]                                    f3 [8]
     'z' (122) |11111101|1 [9]                                  1fb [9]
     '{' (123) |11111111|11111110|1 [17]                     1fffd [17]
     '|' (124) |11111111|1100 [12]                             ffc [12]
     '}' (125) |11111111|11111111|0 [17]                     1fffe [17]
     '~' (126) |11111111|1101 [12]                             ffd [12]
         (127) |11111111|11111111|11111011|101 [27]      7ffffdd [27]
         (128) |11111111|11111111|11111011|110 [27]      7ffffde [27]
         (129) |11111111|11111111|11111011|111 [27]      7ffffdf [27]
         (130) |11111111|11111111|11111100|000 [27]      7ffffe0 [27]
         (131) |11111111|11111111|11111100|001 [27]      7ffffe1 [27]
         (132) |11111111|11111111|11111100|010 [27]      7ffffe2 [27]
         (133) |11111111|11111111|11111100|011 [27]      7ffffe3 [27]
         (134) |11111111|11111111|11111100|100 [27]      7ffffe4 [27]
         (135) |11111111|11111111|11111100|101 [27]      7ffffe5 [27]
         (136) |11111111|11111111|11111100|110 [27]      7ffffe6 [27]
         (137) |11111111|11111111|11111100|111 [27]      7ffffe7 [27]
         (138) |11111111|11111111|11111101|000 [27]      7ffffe8 [27]
         (139) |11111111|11111111|11111101|001 [27]      7ffffe9 [27]
         (140) |11111111|11111111|11111101|010 [27]      7ffffea [27]
         (141) |11111111|11111111|11111101|011 [27]      7ffffeb [27]
         (142) |11111111|11111111|11111101|100 [27]      7ffffec [27]
         (143) |11111111|11111111|11111101|101 [27]      7ffffed [27]
         (144) |11111111|11111111|11111101|110 [27]      7ffffee [27]
         (145) |11111111|11111111|11111101|111 [27]      7fffef [27]
         (146) |11111111|11111111|11111110|000 [27]      7fffff0 [27]
         (147) |11111111|11111111|11111110|001 [27]      7fffff1 [27]
         (148) |11111111|11111111|11111110|010 [27]      7fffff2 [27]
         (149) |11111111|11111111|11111110|011 [27]      7fffff3 [27]
         (150) |11111111|11111111|11111110|100 [27]      7fffff4 [27]
         (151) |11111111|11111111|11111110|101 [27]      7fffff5 [27]
         (152) |11111111|11111111|11111110|110 [27]      7fffff6 [27]
         (153) |11111111|11111111|11111110|111 [27]      7fffff7 [27]
         (154) |11111111|11111111|11111111|000 [27]      7fffff8 [27]
         (155) |11111111|11111111|11111111|001 [27]      7fffff9 [27]
         (156) |11111111|11111111|11111111|010 [27]      7fffffa [27]
         (157) |11111111|11111111|11111111|011 [27]      7fffffb [27]
         (158) |11111111|11111111|11111111|100 [27]      7fffffc [27]
         (159) |11111111|11111111|11111111|101 [27]      7fffffd [27]
         (160) |11111111|11111111|11111111|110 [27]      7fffffe [27]
         (161) |11111111|11111111|11111111|111 [27]      7ffffff [27]
         (162) |11111111|11111111|11100000|00 [26]       3ffff80 [26]
         (163) |11111111|11111111|11100000|01 [26]       3ffff81 [26]
         (164) |11111111|11111111|11100000|10 [26]       3ffff82 [26]
         (165) |11111111|11111111|11100000|11 [26]       3ffff83 [26]
         (166) |11111111|11111111|11100001|00 [26]       3ffff84 [26]
         (167) |11111111|11111111|11100001|01 [26]       3ffff85 [26]
         (168) |11111111|11111111|11100001|10 [26]       3ffff86 [26]
```

```
(169) |11111111|11111111|11100001|11 [26]      3ffff87 [26]
(170) |11111111|11111111|11100010|00 [26]      3ffff88 [26]
(171) |11111111|11111111|11100010|01 [26]      3ffff89 [26]
(172) |11111111|11111111|11100010|10 [26]      3ffff8a [26]
(173) |11111111|11111111|11100010|11 [26]      3ffff8b [26]
(174) |11111111|11111111|11100011|00 [26]      3ffff8c [26]
(175) |11111111|11111111|11100011|01 [26]      3ffff8d [26]
(176) |11111111|11111111|11100011|10 [26]      3ffff8e [26]
(177) |11111111|11111111|11100011|11 [26]      3ffff8f [26]
(178) |11111111|11111111|11100100|00 [26]      3ffff90 [26]
(179) |11111111|11111111|11100100|01 [26]      3ffff91 [26]
(180) |11111111|11111111|11100100|10 [26]      3ffff92 [26]
(181) |11111111|11111111|11100100|11 [26]      3ffff93 [26]
(182) |11111111|11111111|11100101|00 [26]      3ffff94 [26]
(183) |11111111|11111111|11100101|01 [26]      3ffff95 [26]
(184) |11111111|11111111|11100101|10 [26]      3ffff96 [26]
(185) |11111111|11111111|11100101|11 [26]      3ffff97 [26]
(186) |11111111|11111111|11100110|00 [26]      3ffff98 [26]
(187) |11111111|11111111|11100110|01 [26]      3ffff99 [26]
(188) |11111111|11111111|11100110|10 [26]      3ffff9a [26]
(189) |11111111|11111111|11100110|11 [26]      3ffff9b [26]
(190) |11111111|11111111|11100111|00 [26]      3ffff9c [26]
(191) |11111111|11111111|11100111|01 [26]      3ffff9d [26]
(192) |11111111|11111111|11100111|10 [26]      3ffff9e [26]
(193) |11111111|11111111|11100111|11 [26]      3ffff9f [26]
(194) |11111111|11111111|11101000|00 [26]      3ffffa0 [26]
(195) |11111111|11111111|11101000|01 [26]      3ffffa1 [26]
(196) |11111111|11111111|11101000|10 [26]      3ffffa2 [26]
(197) |11111111|11111111|11101000|11 [26]      3ffffa3 [26]
(198) |11111111|11111111|11101001|00 [26]      3ffffa4 [26]
(199) |11111111|11111111|11101001|01 [26]      3ffffa5 [26]
(200) |11111111|11111111|11101001|10 [26]      3ffffa6 [26]
(201) |11111111|11111111|11101001|11 [26]      3ffffa7 [26]
(202) |11111111|11111111|11101010|00 [26]      3ffffa8 [26]
(203) |11111111|11111111|11101010|01 [26]      3ffffa9 [26]
(204) |11111111|11111111|11101010|10 [26]      3ffffaa [26]
(205) |11111111|11111111|11101010|11 [26]      3ffffab [26]
(206) |11111111|11111111|11101011|00 [26]      3ffffac [26]
(207) |11111111|11111111|11101011|01 [26]      3ffffad [26]
(208) |11111111|11111111|11101011|10 [26]      3ffffae [26]
(209) |11111111|11111111|11101011|11 [26]      3ffffaf [26]
(210) |11111111|11111111|11101100|00 [26]      3ffffb0 [26]
(211) |11111111|11111111|11101100|01 [26]      3ffffb1 [26]
(212) |11111111|11111111|11101100|10 [26]      3ffffb2 [26]
(213) |11111111|11111111|11101100|11 [26]      3ffffb3 [26]
(214) |11111111|11111111|11101101|00 [26]      3ffffb4 [26]
(215) |11111111|11111111|11101101|01 [26]      3ffffb5 [26]
(216) |11111111|11111111|11101101|10 [26]      3ffffb6 [26]
```

```
(217) |11111111|11111111|11101101|11 [26]       3ffffb7 [26]
(218) |11111111|11111111|11101110|00 [26]       3ffffb8 [26]
(219) |11111111|11111111|11101110|01 [26]       3ffffb9 [26]
(220) |11111111|11111111|11101110|10 [26]       3ffffba [26]
(221) |11111111|11111111|11101110|11 [26]       3ffffbb [26]
(222) |11111111|11111111|11101111|00 [26]       3ffffbc [26]
(223) |11111111|11111111|11101111|01 [26]       3ffffbd [26]
(224) |11111111|11111111|11101111|10 [26]       3ffffbe [26]
(225) |11111111|11111111|11101111|11 [26]       3ffffbf [26]
(226) |11111111|11111111|11110000|00 [26]       3ffffc0 [26]
(227) |11111111|11111111|11110000|01 [26]       3ffffc1 [26]
(228) |11111111|11111111|11110000|10 [26]       3ffffc2 [26]
(229) |11111111|11111111|11110000|11 [26]       3ffffc3 [26]
(230) |11111111|11111111|11110001|00 [26]       3ffffc4 [26]
(231) |11111111|11111111|11110001|01 [26]       3ffffc5 [26]
(232) |11111111|11111111|11110001|10 [26]       3ffffc6 [26]
(233) |11111111|11111111|11110001|11 [26]       3ffffc7 [26]
(234) |11111111|11111111|11110010|00 [26]       3ffffc8 [26]
(235) |11111111|11111111|11110010|01 [26]       3ffffc9 [26]
(236) |11111111|11111111|11110010|10 [26]       3ffffca [26]
(237) |11111111|11111111|11110010|11 [26]       3ffffcb [26]
(238) |11111111|11111111|11110011|00 [26]       3ffffcc [26]
(239) |11111111|11111111|11110011|01 [26]       3ffffcd [26]
(240) |11111111|11111111|11110011|10 [26]       3ffffce [26]
(241) |11111111|11111111|11110011|11 [26]       3ffffcf [26]
(242) |11111111|11111111|11110100|00 [26]       3ffffd0 [26]
(243) |11111111|11111111|11110100|01 [26]       3ffffd1 [26]
(244) |11111111|11111111|11110100|10 [26]       3ffffd2 [26]
(245) |11111111|11111111|11110100|11 [26]       3ffffd3 [26]
(246) |11111111|11111111|11110101|00 [26]       3ffffd4 [26]
(247) |11111111|11111111|11110101|01 [26]       3ffffd5 [26]
(248) |11111111|11111111|11110101|10 [26]       3ffffd6 [26]
(249) |11111111|11111111|11110101|11 [26]       3ffffd7 [26]
(250) |11111111|11111111|11110110|00 [26]       3ffffd8 [26]
(251) |11111111|11111111|11110110|01 [26]       3ffffd9 [26]
(252) |11111111|11111111|11110110|10 [26]       3ffffda [26]
(253) |11111111|11111111|11110110|11 [26]       3ffffdb [26]
(254) |11111111|11111111|11110111|00 [26]       3ffffdc [26]
(255) |11111111|11111111|11110111|01 [26]       3ffffdd [26]
(256) |10010 [5]                                    12 [5]
```

18. Appendix C

Appendix C huffman code-table for responses

```
              aligned                                   aligned
```

sym	to MSB as bits	len	to LSB as hex	len
(0)	|11111111|11111111|11101111|10	[26]	3ffffbe	[26]
(1)	|11111111|11111111|11101111|11	[26]	3ffffbf	[26]
(2)	|11111111|11111111|11110000|00	[26]	3ffffc0	[26]
(3)	|11111111|11111111|11110000|01	[26]	3ffffc1	[26]
(4)	|11111111|11111111|11110000|10	[26]	3ffffc2	[26]
(5)	|11111111|11111111|11110000|11	[26]	3ffffc3	[26]
(6)	|11111111|11111111|11110001|00	[26]	3ffffc4	[26]
(7)	|11111111|11111111|11110001|01	[26]	3ffffc5	[26]
(8)	|11111111|11111111|11110001|10	[26]	3ffffc6	[26]
(9)	|11111111|11111111|11110001|11	[26]	3ffffc7	[26]
(10)	|11111111|11111111|11110010|00	[26]	3ffffc8	[26]
(11)	|11111111|11111111|11110010|01	[26]	3ffffc9	[26]
(12)	|11111111|11111111|11110010|10	[26]	3ffffca	[26]
(13)	|11111111|11111111|11110010|11	[26]	3ffffcb	[26]
(14)	|11111111|11111111|11110011|00	[26]	3ffffcc	[26]
(15)	|11111111|11111111|11110011|01	[26]	3ffffcd	[26]
(16)	|11111111|11111111|11110011|10	[26]	3ffffce	[26]
(17)	|11111111|11111111|11110011|11	[26]	3ffffcf	[26]
(18)	|11111111|11111111|11110100|00	[26]	3ffffd0	[26]
(19)	|11111111|11111111|11110100|01	[26]	3ffffd1	[26]
(20)	|11111111|11111111|11110100|10	[26]	3ffffd2	[26]
(21)	|11111111|11111111|11110100|11	[26]	3ffffd3	[26]
(22)	|11111111|11111111|11110101|00	[26]	3ffffd4	[26]
(23)	|11111111|11111111|11110101|01	[26]	3ffffd5	[26]
(24)	|11111111|11111111|11110101|10	[26]	3ffffd6	[26]
(25)	|11111111|11111111|11110101|11	[26]	3ffffd7	[26]
(26)	|11111111|11111111|11110110|00	[26]	3ffffd8	[26]
(27)	|11111111|11111111|11110110|01	[26]	3ffffd9	[26]
(28)	|11111111|11111111|11110110|10	[26]	3ffffda	[26]
(29)	|11111111|11111111|11110110|11	[26]	3ffffdb	[26]
(30)	|11111111|11111111|11110111|00	[26]	3ffffdc	[26]
(31)	|11111111|11111111|11110111|01	[26]	3ffffdd	[26]
' ' (32)	|0000	[4]	0	[4]
'!' (33)	|11111111|1010	[12]	ffa	[12]
'"' (34)	|1101000	[7]	68	[7]
'#' (35)	|11111111|111010	[14]	3ffa	[14]
'$' (36)	|11111111|1111100	[15]	7ffc	[15]
'%' (37)	|11110101|0	[9]	1ea	[9]
'&' (38)	|11111110|00	[10]	3f8	[10]
''' (39)	|11111111|11100	[13]	1ffc	[13]
'(' (40)	|11110101|1	[9]	1eb	[9]
')' (41)	|11110110|0	[9]	1ec	[9]
'*' (42)	|11111111|1011	[12]	ffb	[12]
'+' (43)	|11111110|01	[10]	3f9	[10]
',' (44)	|100110	[6]	26	[6]

```
'-' ( 45) |100111 [6]                                  27 [6]
'.' ( 46) |1101001 [7]                                 69 [7]
'/' ( 47) |11101000 [8]                                e8 [8]
'0' ( 48) |0001 [4]                                     1 [4]
'1' ( 49) |0010 [4]                                     2 [4]
'2' ( 50) |0011 [4]                                     3 [4]
'3' ( 51) |01000 [5]                                    8 [5]
'4' ( 52) |01001 [5]                                    9 [5]
'5' ( 53) |01010 [5]                                    a [5]
'6' ( 54) |101000 [6]                                  28 [6]
'7' ( 55) |01011 [5]                                    b [5]
'8' ( 56) |01100 [5]                                    c [5]
'9' ( 57) |01101 [5]                                    d [5]
':' ( 58) |01110 [5]                                    e [5]
';' ( 59) |11110110|1 [9]                             1ed [9]
'<' ( 60) |11111111|11111100 [16]                    fffc [16]
'=' ( 61) |1101010 [7]                                 6a [7]
'>' ( 62) |11111111|111011 [14]                      3ffb [14]
'?' ( 63) |11111111|1100 [12]                         ffc [12]
'@' ( 64) |11111111|11111110|0 [17]                 1fffc [17]
'A' ( 65) |1101011 [7]                                 6b [7]
'B' ( 66) |11110111|0 [9]                             1ee [9]
'C' ( 67) |11101001 [8]                                e9 [8]
'D' ( 68) |11101010 [8]                                ea [8]
'E' ( 69) |11101011 [8]                                eb [8]
'F' ( 70) |11101100 [8]                                ec [8]
'G' ( 71) |101001 [6]                                  29 [6]
'H' ( 72) |11110111|1 [9]                             1ef [9]
'I' ( 73) |11111000|0 [9]                             1f0 [9]
'J' ( 74) |11101101 [8]                                ed [8]
'K' ( 75) |11111110|10 [10]                           3fa [10]
'L' ( 76) |11111000|1 [9]                             1f1 [9]
'M' ( 77) |101010 [6]                                  2a [6]
'N' ( 78) |11101110 [8]                                ee [8]
'O' ( 79) |11101111 [8]                                ef [8]
'P' ( 80) |11111001|0 [9]                             1f2 [9]
'Q' ( 81) |11111001|1 [9]                             1f3 [9]
'R' ( 82) |11111010|0 [9]                             1f4 [9]
'S' ( 83) |1101100 [7]                                 6c [7]
'T' ( 84) |01111 [5]                                    f [5]
'U' ( 85) |11111010|1 [9]                             1f5 [9]
'V' ( 86) |11111011|0 [9]                             1f6 [9]
'W' ( 87) |11110000 [8]                                f0 [8]
'X' ( 88) |11111110|11 [10]                           3fb [10]
'Y' ( 89) |11111111|00 [10]                           3fc [10]
'Z' ( 90) |11111111|01 [10]                           3fd [10]
'[' ( 91) |11111111|1101 [12]                         ffd [12]
'\' ( 92) |11111111|111100 [14]                      3ffc [14]
```

```
']' ( 93) |11111111|100 [11]                                7fc [11]
'^' ( 94) |11111111|1111101 [15]                           7ffd [15]
'_' ( 95) |11111011|1 [9]                                   1f7 [9]
'`' ( 96) |11111111|11111111|10 [18]                      3fffe [18]
'a' ( 97) |10000 [5]                                         10 [5]
'b' ( 98) |1101101 [7]                                       6d [7]
'c' ( 99) |101011 [6]                                        2b [6]
'd' (100) |101100 [6]                                        2c [6]
'e' (101) |10001 [5]                                         11 [5]
'f' (102) |1101110 [7]                                       6e [7]
'g' (103) |1101111 [7]                                       6f [7]
'h' (104) |1110000 [7]                                       70 [7]
'i' (105) |101101 [6]                                        2d [6]
'j' (106) |11111100|0 [9]                                   1f8 [9]
'k' (107) |11111100|1 [9]                                   1f9 [9]
'l' (108) |1110001 [7]                                       71 [7]
'm' (109) |1110010 [7]                                       72 [7]
'n' (110) |101110 [6]                                        2e [6]
'o' (111) |101111 [6]                                        2f [6]
'p' (112) |110000 [6]                                        30 [6]
'q' (113) |11111101|0 [9]                                   1fa [9]
'r' (114) |110001 [6]                                        31 [6]
's' (115) |1110011 [7]                                       73 [7]
't' (116) |110010 [6]                                        32 [6]
'u' (117) |110011 [6]                                        33 [6]
'v' (118) |11110001 [8]                                      f1 [8]
'w' (119) |11110010 [8]                                      f2 [8]
'x' (120) |11110011 [8]                                      f3 [8]
'y' (121) |11110100 [8]                                      f4 [8]
'z' (122) |11111101|1 [9]                                   1fb [9]
'{' (123) |11111111|11111110|1 [17]                       1fffd [17]
'|' (124) |11111111|111101 [14]                            3ffd [14]
'}' (125) |11111111|11111111|0 [17]                       1fffe [17]
'~' (126) |11111111|11111101 [16]                          fffd [16]
    (127) |11111111|11111111|11110111|10 [26]          3ffffde [26]
    (128) |11111111|11111111|11110111|11 [26]          3ffffdf [26]
    (129) |11111111|11111111|11111000|00 [26]          3ffffe0 [26]
    (130) |11111111|11111111|11111000|01 [26]          3ffffe1 [26]
    (131) |11111111|11111111|11111000|10 [26]          3ffffe2 [26]
    (132) |11111111|11111111|11111000|11 [26]          3ffffe3 [26]
    (133) |11111111|11111111|11111001|00 [26]          3ffffe4 [26]
    (134) |11111111|11111111|11111001|01 [26]          3ffffe5 [26]
    (135) |11111111|11111111|11111001|10 [26]          3ffffe6 [26]
    (136) |11111111|11111111|11111001|11 [26]          3ffffe7 [26]
    (137) |11111111|11111111|11111010|00 [26]          3ffffe8 [26]
    (138) |11111111|11111111|11111010|01 [26]          3ffffe9 [26]
    (139) |11111111|11111111|11111010|10 [26]          3ffffea [26]
    (140) |11111111|11111111|11111010|11 [26]          3ffffeb [26]
```

```
(141) |11111111|11111111|11111011|00 [26]       3ffffec [26]
(142) |11111111|11111111|11111011|01 [26]       3fffed [26]
(143) |11111111|11111111|11111011|10 [26]       3ffffee [26]
(144) |11111111|11111111|11111011|11 [26]       3fffef [26]
(145) |11111111|11111111|11111100|00 [26]       3fffff0 [26]
(146) |11111111|11111111|11111100|01 [26]       3fffff1 [26]
(147) |11111111|11111111|11111100|10 [26]       3fffff2 [26]
(148) |11111111|11111111|11111100|11 [26]       3fffff3 [26]
(149) |11111111|11111111|11111101|00 [26]       3fffff4 [26]
(150) |11111111|11111111|11111101|01 [26]       3fffff5 [26]
(151) |11111111|11111111|11111101|10 [26]       3fffff6 [26]
(152) |11111111|11111111|11111101|11 [26]       3fffff7 [26]
(153) |11111111|11111111|11111110|00 [26]       3fffff8 [26]
(154) |11111111|11111111|11111110|01 [26]       3fffff9 [26]
(155) |11111111|11111111|11111110|10 [26]       3fffffa [26]
(156) |11111111|11111111|11111110|11 [26]       3fffffb [26]
(157) |11111111|11111111|11111111|00 [26]       3fffffc [26]
(158) |11111111|11111111|11111111|01 [26]       3fffffd [26]
(159) |11111111|11111111|11111111|10 [26]       3fffffe [26]
(160) |11111111|11111111|11111111|11 [26]       3ffffff [26]
(161) |11111111|11111111|11000000|0 [25]        1ffff80 [25]
(162) |11111111|11111111|11000000|1 [25]        1ffff81 [25]
(163) |11111111|11111111|11000001|0 [25]        1ffff82 [25]
(164) |11111111|11111111|11000001|1 [25]        1ffff83 [25]
(165) |11111111|11111111|11000010|0 [25]        1ffff84 [25]
(166) |11111111|11111111|11000010|1 [25]        1ffff85 [25]
(167) |11111111|11111111|11000011|0 [25]        1ffff86 [25]
(168) |11111111|11111111|11000011|1 [25]        1ffff87 [25]
(169) |11111111|11111111|11000100|0 [25]        1ffff88 [25]
(170) |11111111|11111111|11000100|1 [25]        1ffff89 [25]
(171) |11111111|11111111|11000101|0 [25]        1ffff8a [25]
(172) |11111111|11111111|11000101|1 [25]        1ffff8b [25]
(173) |11111111|11111111|11000110|0 [25]        1ffff8c [25]
(174) |11111111|11111111|11000110|1 [25]        1ffff8d [25]
(175) |11111111|11111111|11000111|0 [25]        1ffff8e [25]
(176) |11111111|11111111|11000111|1 [25]        1ffff8f [25]
(177) |11111111|11111111|11001000|0 [25]        1ffff90 [25]
(178) |11111111|11111111|11001000|1 [25]        1ffff91 [25]
(179) |11111111|11111111|11001001|0 [25]        1ffff92 [25]
(180) |11111111|11111111|11001001|1 [25]        1ffff93 [25]
(181) |11111111|11111111|11001010|0 [25]        1ffff94 [25]
(182) |11111111|11111111|11001010|1 [25]        1ffff95 [25]
(183) |11111111|11111111|11001011|0 [25]        1ffff96 [25]
(184) |11111111|11111111|11001011|1 [25]        1ffff97 [25]
(185) |11111111|11111111|11001100|0 [25]        1ffff98 [25]
(186) |11111111|11111111|11001100|1 [25]        1ffff99 [25]
(187) |11111111|11111111|11001101|0 [25]        1ffff9a [25]
(188) |11111111|11111111|11001101|1 [25]        1ffff9b [25]
```

```
(189) |11111111|11111111|11001110|0 [25]       1ffff9c [25]
(190) |11111111|11111111|11001110|1 [25]       1ffff9d [25]
(191) |11111111|11111111|11001111|0 [25]       1ffff9e [25]
(192) |11111111|11111111|11001111|1 [25]       1ffff9f [25]
(193) |11111111|11111111|11010000|0 [25]       1ffffa0 [25]
(194) |11111111|11111111|11010000|1 [25]       1ffffa1 [25]
(195) |11111111|11111111|11010001|0 [25]       1ffffa2 [25]
(196) |11111111|11111111|11010001|1 [25]       1ffffa3 [25]
(197) |11111111|11111111|11010010|0 [25]       1ffffa4 [25]
(198) |11111111|11111111|11010010|1 [25]       1ffffa5 [25]
(199) |11111111|11111111|11010011|0 [25]       1ffffa6 [25]
(200) |11111111|11111111|11010011|1 [25]       1ffffa7 [25]
(201) |11111111|11111111|11010100|0 [25]       1ffffa8 [25]
(202) |11111111|11111111|11010100|1 [25]       1ffffa9 [25]
(203) |11111111|11111111|11010101|0 [25]       1ffffaa [25]
(204) |11111111|11111111|11010101|1 [25]       1ffffab [25]
(205) |11111111|11111111|11010110|0 [25]       1ffffac [25]
(206) |11111111|11111111|11010110|1 [25]       1ffffad [25]
(207) |11111111|11111111|11010111|0 [25]       1ffffae [25]
(208) |11111111|11111111|11010111|1 [25]       1ffffaf [25]
(209) |11111111|11111111|11011000|0 [25]       1ffffb0 [25]
(210) |11111111|11111111|11011000|1 [25]       1ffffb1 [25]
(211) |11111111|11111111|11011001|0 [25]       1ffffb2 [25]
(212) |11111111|11111111|11011001|1 [25]       1ffffb3 [25]
(213) |11111111|11111111|11011010|0 [25]       1ffffb4 [25]
(214) |11111111|11111111|11011010|1 [25]       1ffffb5 [25]
(215) |11111111|11111111|11011011|0 [25]       1ffffb6 [25]
(216) |11111111|11111111|11011011|1 [25]       1ffffb7 [25]
(217) |11111111|11111111|11011100|0 [25]       1ffffb8 [25]
(218) |11111111|11111111|11011100|1 [25]       1ffffb9 [25]
(219) |11111111|11111111|11011101|0 [25]       1ffffba [25]
(220) |11111111|11111111|11011101|1 [25]       1ffffbb [25]
(221) |11111111|11111111|11011110|0 [25]       1ffffbc [25]
(222) |11111111|11111111|11011110|1 [25]       1ffffbd [25]
(223) |11111111|11111111|11011111|0 [25]       1ffffbe [25]
(224) |11111111|11111111|11011111|1 [25]       1ffffbf [25]
(225) |11111111|11111111|11100000|0 [25]       1ffffc0 [25]
(226) |11111111|11111111|11100000|1 [25]       1ffffc1 [25]
(227) |11111111|11111111|11100001|0 [25]       1ffffc2 [25]
(228) |11111111|11111111|11100001|1 [25]       1ffffc3 [25]
(229) |11111111|11111111|11100010|0 [25]       1ffffc4 [25]
(230) |11111111|11111111|11100010|1 [25]       1ffffc5 [25]
(231) |11111111|11111111|11100011|0 [25]       1ffffc6 [25]
(232) |11111111|11111111|11100011|1 [25]       1ffffc7 [25]
(233) |11111111|11111111|11100100|0 [25]       1ffffc8 [25]
(234) |11111111|11111111|11100100|1 [25]       1ffffc9 [25]
(235) |11111111|11111111|11100101|0 [25]       1ffffca [25]
(236) |11111111|11111111|11100101|1 [25]       1ffffcb [25]
```

```
(237) |11111111|11111111|11100110|0 [25]       1ffffcc [25]
(238) |11111111|11111111|11100110|1 [25]       1ffffcd [25]
(239) |11111111|11111111|11100111|0 [25]       1ffffce [25]
(240) |11111111|11111111|11100111|1 [25]       1ffffcf [25]
(241) |11111111|11111111|11101000|0 [25]       1ffffd0 [25]
(242) |11111111|11111111|11101000|1 [25]       1ffffd1 [25]
(243) |11111111|11111111|11101001|0 [25]       1ffffd2 [25]
(244) |11111111|11111111|11101001|1 [25]       1ffffd3 [25]
(245) |11111111|11111111|11101010|0 [25]       1ffffd4 [25]
(246) |11111111|11111111|11101010|1 [25]       1ffffd5 [25]
(247) |11111111|11111111|11101011|0 [25]       1ffffd6 [25]
(248) |11111111|11111111|11101011|1 [25]       1ffffd7 [25]
(249) |11111111|11111111|11101100|0 [25]       1ffffd8 [25]
(250) |11111111|11111111|11101100|1 [25]       1ffffd9 [25]
(251) |11111111|11111111|11101101|0 [25]       1ffffda [25]
(252) |11111111|11111111|11101101|1 [25]       1ffffdb [25]
(253) |11111111|11111111|11101110|0 [25]       1ffffdc [25]
(254) |11111111|11111111|11101110|1 [25]       1ffffdd [25]
(255) |11111111|11111111|11101111|0 [25]       1ffffde [25]
(256) |10010 [5]                                    12 [5]
```

19. Normative References

[CANON] Schwartz, E. S. and Kallick, B., "Generating a canonical prefix encoding", Comm. ACM, 7,3 (March 1964), pp 166-169 .

[HUFF] Huffman, D. A., "A Method for the Construction of Minimim Redundancy Codes", Proceedings of the Institute of Radio Engineers, September 1952, Volume 40, Number 9, pp. 1098-1101 .

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[SPDY] Belshe, M. and R. Peon, "SPDY PROTOCOL", <http://tools.ietf.org/html/draft-mbelshe-httpbis-spdy>.

Author's Address

Roberto Peon
Google, Inc

Email: fenix@google.com