SIP Compression using ITU-T V.44 with Pre-loaded Dictionary

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and it working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Status of this Memo

This memo is an Internet-Draft that provides information for the Internet community. Distribution of this memo is unlimited.

The file name for this Internet-Draft is draft-heath-rohc-sip-v44-00.txt. Comments are invited and should be addressed to the author whose contact information is in Section 8.

Abstract

This document describes a compression method based on the data compression algorithm described in ITU-T Recommendation V.44 [V44]. Recommendation V.44 is a modem standard but Annex B of the recommendation describes the implementation of V.44 in packet networks. This document defines the application of V.44 to packets or messages where some or all of the contents of the data is known ahead of time. For such applications the V.44 dictionary is partially pre-loaded after initialization with words and/or phrases that are expected to appear in the data. This Internet Draft describes the mechanism of the dictionary pre-load and its advantages for applications such as SIP/SDP.

The V.44 algorithm is unique among data compression algorithms in
that it combines excellent compression ratios with very fast 
execution times and low memory requirements. It learns and adapts 
to the input data very quickly which is one reason it provides 
superior compression ratios on relatively small blocks of data. 
The partial pre-load of the dictionary with expected words and 
phrases will allow even better compression ratios since it will 
allow V.44 to encode each pre-loaded word or phrase with between 
7 to 10 bits each time it is encountered in the data.

V.44 is lossless and is based upon the LZJH data compression 
algorithm. As described in Annex B of [V44], LZJH operates in 
packet networks using one of two methods: Packet Method where each 
individual IP packet is compressed/decompressed separately, or 
Multi-Packet Method where several IP packets are 
compressed/decompressed as a continuation using a reliable transport.

Thoughout the remainder of this document the terms V.44 and LZJH 
are synonomous.

Table of Contents

1. Introduction...................................................2
   1.1 Advantages of V.44 Data Compression......................3
   1.2 Background of LZJH Compression..........................4
2. LZJH Design and Operation for SIP.............................4
   2.1 LZJH Modifications for Pre-loaded Dictionaries.........5
   2.2 Operation and Performance................................6
   2.3 Data Integrity............................................7
3. LZJH Pre-load Test Results....................................7
   3.1 Test 1....................................................7
   3.2 Test 1A...................................................7
   3.3 Test 2....................................................7
   3.4 Test 3....................................................8
   3.5 Test Conclusions..........................................8
4. Algorithm Comparison...........................................9
   4.1 Comparison Example 1......................................9
   4.2 Comparison Example 2......................................9
   4.3 Comparison Example 3......................................9
   4.4 Comparison Conclusions....................................10
5. Security Considerations.......................................10
6. Intellectual Property Right Considerations..................10
7. References..................................................................10
8. Authors’ Address.................................................11
9. Full Copyright Statement.......................................11

1. Introduction

Cellular and wireless networks are planning on implementing more 
and more Internet, IP based, services in the future. One problem 
with the set of IP protocols invented with the Internet in mind is 
the high overhead of additional headers and verbose data fields. 
Since bandwidth is at a premium in cellular, satellite, and other
wireless networks the future implementation of these protocols poses a bandwidth efficiency problem.

For example, application signaling protocols such as SIP [SIP], SDP [SDP] and RTSP [RTSP] will typically be used to setup and control applications in a mobile Internet based on a cellular network. However, the generous size of the protocol messages combined with their request/response nature create delays and waste bandwidth in a cellular network.

In order to reduce the delays incurred and bandwidth used by these protocols it is necessary to use compression techniques in cellular, satellite, and other wireless networks. Some of these protocols contain messages that contain data that is known in advance and is highly redundant. Since the same protocols also have messages containing ASCII free form text, a general purpose data compression algorithm which can take advantage of words or phrases known to appear in the text is the best solution.

1.1 Advantages of V.44 Data Compression

Below are some of the reasons ITU-T V.44 data compression should be considered for this environment:

A. The primary reason for using compression for SIP/SDP is to reduce the amount of data going over the RAN. LZJH gets the best compression in this environment and therefore will be the most effective in data reduction.

B. LZJH is a general purpose algorithm. If operating in a network entity for one application, such as SIP/SDP, it can be used for other applications such as the compression of IP data packets [IPCOMP] to improve the performance of the Radio Access Network.

C. LZJH is the basis of the latest ITU-T data compression standard, ITU-T V.44 and will become widely used in networks. New developments, such as SIP/SDP in UMTS, will benefit from using the latest standarized compression technology.

D. As defined in Annex B of V.44, the algorithm operates in packet networks on one packet at a time or on several packets using a reliable transport. LZJH source code handles both methods on a per session basis at run time. Thus, each network entity can independently determine, based on its resources, which method to use at any time.

E. LZJH is superior to the other algorithms being considered for the SIP/SDP type application. In this application, LZJH will get about twice the compression as LZW using the same MIPS and memory. LZJH will get 20% to 35% better compression than LZSS while using about one fifth the MIPS and similar memory. Refer to section 4 for algorithm comparisons.
F. LZJH was designed for packetized data. It can get good compression on packets of 200 bytes and decent compression on packets as small as 50 bytes.

1.2 Background of LZJH Compression

LZJH is similar to the algorithm described in [LZ78] although is has aspects which are similar to the algorithm described in [LZ77]. As such, it provides the execution speed and low memory requirements of [LZ78] with compression ratios that are better than [LZ77]. Originally developed for the satellite industry to compress IP datagrams independently, it is ideal for the SIP/SDP application. The LZJH algorithm was modified to compress a continuous stream of data for a modem environment and this modified version is the basis for Recommendation V.44. LZJH is an adaptive, general purpose, lossless data compression algorithm that provides excellent performance across a wide variety of data types, particularly ASCII text with high redundancy.

A typical [LZ78] compression algorithm, such as LZW, is not suitable for application in a packet network since it takes too long to build up its dictionary, resulting in poor compression ratios on IP datagrams that are compressed separately.

A typical [LZ77] compression algorithm, such as LZSS, suffers in the packet network application due to poor execution times.

LZJH not only has superior compression ratios when encoding or decoding packetized data, but it uses little memory and provides very fast execution times.

The details of LZJH compression can be found in [V44].

2. LZJH Design and Operation for SIP

As stated earlier, LZJH operates using one of two methods in a packet network. Packet Method where each IP packet, or protocol message, is compressed separately and the dictionaries are re-initialized between messages. Multi-Packet Method where several messages are compressed and data from previous messages remains in the dictionary for use in compressing the current message.

Multi-Packet Method requires that the messages, or packets, be delivered in order without any packet loss. In other words, it requires a reliable transport. The definition of that transport in the SIP/SDP or other similar application is beyond the scope of this document.

Multi-Packet Method requires a separate history buffer to hold the data from the messages that were previously processed. This method also requires that each separate thread or session supported by a network entity, such as the P-CSCF, have a separate
context for compression. This context includes information for the reliable link and compression dictionaries, including history buffer.

In contrast, Packet Method does not require a reliable transport, does not require in-order delivery of messages, and does not require a separate history. Packet Method is implemented in a serving entity, supporting hundreds of simultaneous sessions, with a single instance of encoder and decoder dictionaries. For this application the total memory requirement for both encoder and decoder is a trivial 16K. About 11K for the encoder, and about 5K for the decoder.

Multi-Packet Method provides better compression ratios than Packet Method, but at the cost of considerable memory for a serving network entity. In addition, the cost of complexity and bandwidth to implement the underlying reliable transport may offset the better compression ratios.

2.1 LZJH Modifications for Pre-loaded Dictionaries

The modifications required to pre-load the LZJH dictionaries are minimal and will not affect the basic algorithm. Thus, the same LZJH object module operating in a network entity will be able to operate normally for general purpose compression or operate using pre-loaded dictionaries.

Currently the LZJH algorithm consists of several functions which are called by a user supplied control function which controls the operation of the algorithm. They are:
- initialize Packet Method
- initialize Multi-Packet Method
- re-initialize compressor (encoder) dictionary
- re-initialize decompressor (decoder) dictionary
- compress packet
- decompress packet

New functions to support the pre-load of the dictionaries have been added as follows:
- pre-load compressor dictionary
- pre-load decompressor dictionary
- re-initialize compressor pre-loaded dictionary
- re-initialize decompressor pre-loaded dictionary

The dictionary pre-load procedures will take as an input into the procedure two arrays that define the words or phrases (strings) to be pre-loaded. Note that the encoder and its peer decoder must be pre-loaded with exactly the same arrays for compression/decompression to work correctly. The content, origin, and distribution of the pre-load arrays is beyond the scope of this document and there could conceivably be different pre-load arrays for each message type.

The pre-load arrays consists of an array of the strings to be
pre-loaded and an array of string lengths corresponding to the strings as follows:

```c
char pl_data[] = "INVITE"
    "SIP"
    "/2.0"
    "sip:
    "Via: 
    "SIP/2.0/UDP"
    "From: 
    "To: 
    "Call-ID: 
    "CSeq: 

byte pl_len[] = {6,3,4,4,5,11,6,4,9,6,0};
```

The above defines 10 strings to be pre-loaded. Note that a string length of zero indicates the end of the strings. Also note this mechanism is subject to change as better ideas come along.

Once the encoder and decoder dictionaries are pre-loaded with a set of strings, the re-initialize pre-loaded dictionary procedures will re-initialize only the dynamic portion of the dictionaries and histories, the static, pre-loaded, strings will be maintained. Thus, the LZJH encoder does not have to keep compressing the same pre-loaded data on each message which saves processing time. To remove the pre-loaded information, the control function uses either the re-initialize compressor (or decompressor) function to clear the information or uses the pre-load compressor (or decompressor) dictionary functions to pre-load different information.

It may be desirable to place a byte into the message header or as the first byte of compressed message data to indicate which pre-load table was used by the compressor. Alternatively, the type of message, INVITE, etc., can explicitly indicate which pre-load table was used.

### 2.2 Operation and Performance

During the pre-load of either the compressor (encoder) or decompressor (decoder) dictionary, the information required to define the strings is loaded into the dictionary structures and the strings themselves are concatenated into one array of data that occupies the first N locations of the packet buffer to be processed. Thus, the data to be compressed is copied into the first location following the "static" data in the packet buffer prior to compression.

Once pre-loaded, the dictionary can be re-initialized while maintaining the pre-loaded portion of the dictionary and static data, a major savings in execution speed. With LZSS, it will be necessary to compress the static data each time to rebuild the binary tree.
2.3 Data Integrity

If a CRC or other such mechanism is not used by the underlying link layer to insure the integrity of the messages transferred over the RAN, then it may be necessary to prepend a 16 bit CRC or LRC to the compressed data to insure its integrity prior to decompression.

3. LZJH Pre-load Test Results

The LZJH algorithm was modified as described in section 2 and tests were run on the SIP messages described in section 3.1.1 of <draft-ietf-sip-call-flows-05.txt> [FLOWS].

3.1 Test 1

The first INVITE message and ACK message sent from User A to User B in section 3.1.1 of [FLOWS] are compressed separately with the LZJH dictionary pre-loaded with expected strings.

<table>
<thead>
<tr>
<th>message</th>
<th>uncompressed bytes</th>
<th>compressed bytes</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>423</td>
<td>152</td>
<td>2.8</td>
</tr>
<tr>
<td>ACK</td>
<td>212</td>
<td>88</td>
<td>2.4</td>
</tr>
</tbody>
</table>

Note in section 4.1 the LZJH algorithm without a pre-load reduced the same INVITE message to 266 bytes. The pre-load improved the compression ratio by 75% on the INVITE.

This is the compression ratio on an INVITE and ACK messages compressed separately, using LZJH Packet Method, where a reliable transport or any other mechanism to maintain data between messages is not required. Both used the same set of pre-loaded strings.

3.2 Test 1A

The same INVITE message and ACK message from Test 1 are compressed using Multi-Packet Method where the dictionary is maintained between the two messages.

<table>
<thead>
<tr>
<th>message</th>
<th>uncompressed bytes</th>
<th>compressed bytes</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>423</td>
<td>152</td>
<td>2.8</td>
</tr>
<tr>
<td>ACK</td>
<td>212</td>
<td>29</td>
<td>7.3</td>
</tr>
</tbody>
</table>

3.3 Test 2

The first INVITE message in section 3.1.1 of [FLOWS] is repeated 5 times with certain fields changed for each repetition, such as To URL and display name, rtpmap, etc. This is to provide a ballpark comparison with the "friendly" test in section 5 of UDPCOMP SIP compression [UDP].
As can be seen, compression ratio improvements in the first 2 INVITES are dramatic compared to UDPCOMP and even the last 3 INVITES are better even though the INVITE messages are larger. The total compression ratio of all 5 INVITES is 6.7 compared to the total ratio of 2.7 (2009 / 737) for UDPCOMP, more than double.

This test shows the compression ratio on a series of 5 INVITE messages using LZJH Multi-Packet Method over a reliable transport.

### 3.4 Test 3

Each of the 5 INVITE messages compressed in Test 2, above, is compressed separately using LZJH Packet Method without a reliable transport or any other mechanism to maintain dictionaries between each message.

<table>
<thead>
<tr>
<th>message</th>
<th>uncompressed bytes</th>
<th>compressed bytes</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE 1</td>
<td>423</td>
<td>152</td>
<td>2.8</td>
</tr>
<tr>
<td>INVITE 2</td>
<td>418</td>
<td>48</td>
<td>8.7</td>
</tr>
<tr>
<td>INVITE 3</td>
<td>426</td>
<td>43</td>
<td>9.9</td>
</tr>
<tr>
<td>INVITE 4</td>
<td>425</td>
<td>37</td>
<td>11.5</td>
</tr>
<tr>
<td>INVITE 5</td>
<td>429</td>
<td>37</td>
<td>11.6</td>
</tr>
</tbody>
</table>

The average of the 5 INVITES is a compression ratio of 2.7 which is exactly the same total compression ratio achieved by UDPCOMP on 5 INVITES. Each compressed message stands on its own and will obtain the same compression regardless if previous messages are received.

### 3.5 Test Conclusions

In an environment where the codebook or dictionary of SIP messages is maintained between messages via a reliable transport or other mechanism, LZJH with a pre-loaded dictionary can obtain compression ratios that are more than double those of UDPCOMP (refer to Test 2).

In environment where each message stands on its own, LZJH with a pre-loaded dictionary can reduce a SIP message by some 60% where UDPCOMP actually expands the first INVITE message in [UDP] section 5.

SIP signalling requires the exchange of just a few messages, if it takes a few messages for a complicated mechanism to obtain good compression ratios over the total of all messages then any advantage is negated.
LZJH can obtain very good compression ratios using Multi-Packet Method over a reliable transport. However, since LZJH can reduce SIP messages by more than half without any mechanism where the data in one message is used to compress following messages, the complication of a reliable transport or other such mechanisms may offset the extra savings. Since LZJH supports both methods, it is up to network designers to determine which method to use on each call flow.

4. Algorithm Comparison

Compression ratio comparisons between three algorithms, LZJH, LZSS, and LZW, are provided in this section.

4.1 Comparison Example 1

The first INVITE message in section 3.1.1 of [FLOWS] is compressed by each algorithm without the pre-load of any algorithm. This shows the raw ability of the algorithm to compress a single SIP message.

<table>
<thead>
<tr>
<th>algorithm</th>
<th>uncompressed bytes</th>
<th>compressed bytes</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>LZJH</td>
<td>423</td>
<td>266</td>
<td>-</td>
</tr>
<tr>
<td>LZSS</td>
<td>423</td>
<td>329</td>
<td>23.7%</td>
</tr>
<tr>
<td>LZW</td>
<td>423</td>
<td>329</td>
<td>23.7%</td>
</tr>
</tbody>
</table>

In the above example, LZJH obtains a compression ratio of 1.55 on a single SIP message without a pre-load of the dictionary.

4.2 Comparison Example 2

The first INVITE message in section 3.1.1 of [FLOWS] is duplicated into a single message and then compressed by each algorithm without the pre-load of any algorithm. This compares the ability of the algorithm to compress a SIP message when the same SIP message is already in the dictionary or history memory.

<table>
<thead>
<tr>
<th>algorithm</th>
<th>uncompressed bytes</th>
<th>compressed bytes</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>LZJH</td>
<td>846</td>
<td>273</td>
<td>-</td>
</tr>
<tr>
<td>LZSS</td>
<td>846</td>
<td>357</td>
<td>30.8%</td>
</tr>
<tr>
<td>LZW</td>
<td>846</td>
<td>536</td>
<td>96.3%</td>
</tr>
</tbody>
</table>

This example shows that LZJH uses just 7 additional bytes to encode the entire duplicated 2nd SIP message (273 - 266 = 7). LZSS uses 28 additional bytes, and LZW 207 additional bytes.

4.3 Comparison Example 3

An HTML file with significant redundancy is compressed. This is to test both the compression ratio and encoder speed of each algorithm. The SIP messages are too small for accurate encoder speed measurements. The file is compressed as separate packets of 1500...
bytes each with compression dictionaries re-initialized between each packet, the compression ratio is the average for all packets. The encoder time is total for all packets.

Note that at this writing, the author does not have a version of LZSS using binary trees for speed measurement, thus, a variant of LZSS is used for the speed measurements, Lzs which uses hash tables and is faster than LZSS. Compression ratio measurements are LZSS.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Encoder Time</th>
<th>Uncompressed Bytes</th>
<th>Compressed Bytes</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>LZJH</td>
<td>2</td>
<td>97030</td>
<td>35081</td>
<td>-</td>
</tr>
<tr>
<td>Lzs/LZSS</td>
<td>13</td>
<td>97030</td>
<td>42562</td>
<td>21.3%</td>
</tr>
<tr>
<td>LZW</td>
<td>3</td>
<td>97030</td>
<td>62586</td>
<td>78.4%</td>
</tr>
</tbody>
</table>

The above example shows that both LZJH and LZW encoders are several times faster than that of LZSS. Decoders are not measured here, however, previous testing has shown that the LZJH decoder is about 3 times faster than the LZJH encoder. The LZSS decoder should be as fast as the LZJH decoder and the LZW decoder should be slower by about a factor of 2.

4.4 Comparison Conclusions

Among the 3 algorithms, LZJH is the obvious choice for both speed and compression performance.

5. Security Considerations

There are no specific security considerations regarding the proposal in this Internet Draft. However, the application of data compression, as described herein, should always be done at a layer above any encryption, such as IPSec.

6. Intellectual Property Right Considerations

The LZJH data compression algorithm is referenced in one or more patents or patent applications. To the extent that some of the concepts in this document are adopted in a specification, Hughes Network Systems agrees to license patents technically necessary to implement the specification on fair, reasonable, and nondiscriminatory terms. Since HNS is a potential manufacturer of 3rd generation satellite terminals, this may be based on reciprocity where possible.

7. References


[LZ77] Lempel, A., and Ziv, J., "A Universal Algorithm for


8. Authors’ Address

Jeff Heath
Hughes Network Systems
10450 Pacific Center Ct.
San Diego, CA  92121

voice: 858-452-4826
fax: 858-597-8979
e-mail: jheath@hns.com

9. Full Copyright Statement

Copyright (C) The Internet Society (1998). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an
"AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.