IP Payload Compression Using DEFLATE
<draft-ietf-ippcp-deflate-02.txt>

Status of this Memo

   This document is a submission to the IETF Internet Protocol Payload
   Compression Protocol (IPPCP) Working Group. Comments are solicited
   and should be addressed to the working group mailing list or to the
   editor.

   This document is an Internet-Draft. Internet Drafts are working
   documents of the Internet Engineering Task Force (IETF), its areas,
   and its working Groups. Note that other groups may also distribute
   working documents as Internet Drafts.

   Internet-Drafts draft documents are valid for a maximum of six
   months and may be updated, replaced, or obsolete 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."

   To learn the current status of any Internet-Draft, please check the
   "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow
   Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe),
   munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or
   ftp.isi.edu (US West Coast).

   Distribution of this memo is unlimited.

Abstract

   This document describes a compression method based on the DEFLATE
   compression algorithm. This document defines the application of
   the DEFLATE algorithm to the IP Payload Compression Protocol.

Table of Contents

1. Introduction

The IP Payload Compression Protocol allows the compression of IP
datagrams by supporting different compression algorithms. This
document describes how to integrate the DEFLATE compression
algorithm [Deutsch96] into IPCOMP [IPCOMP].

This document SHOULD be read in conjunction with [IPCOMP] and MUST
be taken in its context.

1.1 The DEFLATE Compression Algorithm

The 'deflate' compression format [Deutsch96], as used by the PKZIP
and gzip compressors and as embodied in the freely and widely
distributed zlib [Gailly95] library source code, has the following
features:

o an apparently unencumbered encoding and compression algorithm,
  with an open and publicly-available specification.

o low-overhead escape mechanism for incompressible data. The PPP
  Deflate specification offers options to reduce that overhead
  further.

o heavily used for many years in networks, on modem and other
  point-to-point links to transfer files for personal computers
  and workstations.

o easily achieves 2:1 compression on the Calgary corpus[Corpus90]
  using less than 64KBytes of memory on both sender and receive.

1.2 **Licensing**

   The zlib source is widely and freely available, subject to the
   following copyright:

      (C) 1995 Jean-Loup Gailly and Mark Adler

      This software is provided 'as-is',without any express or implied
      warranty. In no event will the authors be held liable for any
      damages arising from the use of this software.

      Permission is granted to anyone to use this software for any
      purpose, including commercial applications, and to alter it and
      redistribute it freely, subject to the following restrictions:

      1. The origin of this software must not be misrepresented; you
         must not claim that you wrote the original software. If you
         use this software in a product, an acknowledgment in the
         product documentation would be appreciated but is not
         required.

      2. Altered source versions must be plainly marked as such, and
         must not be misrepresented as being the original software.

      3. This notice may not be removed or altered from any source
         distribution.

         Jean-Loup Gailly   Mark Adler
         gzip@prep.ai.mit.edu   madler@alumni.caltech.edu

      If you use the zlib library in a product, we would appreciate
      not receiving lengthy legal documents to sign. The sources are
      provided for free but without warranty of any kind. The library
      has been entirely written by Jean-Loup Gailly and Mark Adler; it
      does not include third-party code.

   The deflate format and compression algorithm are based on Lempel-
   Ziv LZ77 compression; extensive research has been done by the GNU
   Project and the Portable Network Graphics working group supporting
   its patent free status.

1.3 **Specification of Requirements**

   The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD
   NOT", and "MAY" that appear in this document are to be interpreted
   as described in [Bradner97].

2. **DEFLATE Algorithm Implementation**

   The DEFLATE compression algorithm was designed by Phil Katz and its
   details are publicly available in [Deutsch96]. Thus it is a good
   freely available algorithm to implement within IPCOMP.

   Compression and decompression algorithm details should be followed
   as outlined in [Deutsch96] or the use of a software library may be
   preferable.

2.1 **Compression**

   As defined in [IPCOMP], the compression process is determined by
   the IP Compression Association (IPCA). The IPCA MUST define the
   DEFLATE algorithm for the process within this document to take
   place.

   The compression process entails compressing the data from the IP
   datagram and placing the result after the IPComp header. For
   example, compressing a TCP datagram;

   Before: IP TCP ...
   After: IP IPCOMP (TCP ...)

   Please note how everything after the IPCOMP header is compressed.

   DEFLATE allows for a number of compression levels ranging from best
   compression but slow to fast compression. The level that one
   compresses data is implementation dependant since it is always
   compatible with the decompression algorithm.

2.2 **Decompression**

   As in the compression process, the IPCA defines the parameters and
   algorithm to utilize for the decompression process.

   As defined in [IPCOMP] the data after the IPComp header is
   decompressed and replaces the IPComp header within the IP header.

   Decompression using the DEFLATE algorithm follows the decompression
   process defined in [Deutsch96].

3. Thresholds

   As stated in [IPCOMP], small buffers do not compress well or at all
   as well as fast links since the time it takes to compress is slower
   than the time to transport the data. Informal tests show that the
   average buffer size that produces larger results is around 90
   bytes. Thus implementations SHOULD NOT attempt to compress buffers
   smaller than 90 bytes.

4. IPSec Transform Identifier

   [IPDOI] states that the ISAKMP IPCOMP transform ID for the DEFLATE
   compression algorithm is 2. No other ISAKMP parameters are
   required for the IPCOMP DEFLATE algorithm.

5. Security Considerations

   This document does not add any further security considerations that
   [IPCOMP] and [Deutsch96] have already declared.

6. References

   [IPCOMP]   Shacham, A., Monsour, R., Pereira, R., Thomas, M., "IP
                Payload Compression Protocol (IPComp)", draft-ietf-
                ippcp-protocol-04

   [Deutsch96] P. Deutsch, "DEFLATE Compressed Data Format
                Specification version 1.3", RFC1951, May 1996

   [IPDOI]   Pipper, D. "The Internet IP Security Domain of
                Interpretation for ISAKMP", draft-ietf-ipsec-ipsec-doi-
                06

   [Corpus90]   Bell, T.C., Cleary, G. G. and Witten, I.H., "Text
                Compression", Prentice_Hall, Englewood Cliffs NJ, 1990.
                The compression corpus itself can be found in
                ftp://ftp.uu.net/pub/archiving/zip/

   [Gailly95]   Gailly, J.-L., "Zlib 0.95 beta"

7. Acknowledgments

   The author wishes to thank all of the active members of the IPPCP
   working group especially Abraham Shacham and Naganand Dorswamy.

8. Editor's Address

    Roy Pereira
    <rpereira@timestep.com>
    TimeStep Corporation
    +1 (613) 599-3610 x 4808

   The IP Payload Compression Protocol (IPPCP) working group can be
   contacted via email (ippcp@cisco.com) or through its chair:

    Naganand Dorswamy
    Bay Networks
    <naganand@baynetworks.com>