Network Working Group
Request for Comments: 1979
Category: Informational

J. Woods
Proteon, Inc.

August 1996

PPP Deflate Protocol

Status of This Memo

Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links.

The PPP Compression Control Protocol [2] provides a method to negotiate and utilize compression protocols over PPP encapsulated links.

This document describes the use of the PPP Deflate compression protocol for compressing PPP encapsulated packets.

Table of Contents

1. Introduction

The 'deflate' compression format[3], as used by the PKZIP and gzip compressors and as embodied in the freely and widely distributed zlib[4] library source code, has the following features:

- an apparently unencumbered encoding and compression algorithm, with an open and publically-available specification.
- low-overhead escape mechanism for incompressible data. The PPP Deflate specification offers options to reduce that overhead further.
- heavily used for many years in networks, on modem and other point-to-point links to transfer files for personal computers and workstations.
- easily achieves 2:1 compression on the Calgary corpus[5] using less than 64KBytes of memory on both sender and receive.

1.1. 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.

2. PPP Deflate Packets

Before any PPP Deflate packets may be communicated, PPP must reach the Network-Layer Protocol phase, and the CCP Control Protocol must reach the Opened state.

Exactly one PPP Deflate datagram is encapsulated in the PPP Information field, where the PPP Protocol field contains 0xFD or 0xFB. 0xFD is used when the PPP multilink protocol is not used or "above" multilink. 0xFB is used "below" multilink, to compress independently on individual links of a multilink bundle.

The maximum length of the PPP Deflate datagram transmitted over a PPP link is the same as the maximum length of the Information field of a PPP encapsulated packet.

Only packets with PPP Protocol numbers in the range 0x0000 to 0x3FFF and neither 0xFD nor 0xFB are compressed. Other PPP packets are always sent uncompressed. Control packets are infrequent and should not be compressed for robustness.

Padding

PPP Deflate packets require the previous negotiation of the Self-Describing-Padding Configuration Option [6] if padding is added to packets. If no padding is added, than Self-Describing-Padding is not required.

Reliability and Sequencing

PPP Deflate requires the packets to be delivered in sequence. It relies on Reset-Request and Reset-Ack LCP packets or on renegotiation of the Compression Control Protocol [2] to indicate loss of synchronization between the transmitter and receiver. The LCP FCS detects corrupted packets and the normal mechanisms discard them. Missing or out of order packets are detected by the sequence number in each packet. The packet sequence number ought to be checked before decoding the packet.

Instead of transmitting a Reset-Request packet when detecting a sequence error, the receiver MAY momentarily force CCP to drop out of the Opened state by transmitting a new CCP Configure-Request. This method is more expensive than using Reset-Requests.

When the receiver first encounters an unexpected sequence number it SHOULD send a Reset-Request LCP packet as defined in the Compression Control Protocol. When the transmitter sends the Reset-Ack or when the receiver receives a Reset-ACK, they must reset the sequence number to zero, clear the compression dictionary, and resume sending and receiving compressed packets. The receiver MUST discard all compressed packets after detecting an error and until it receives a Reset-Ack. This strategy can be thought of as abandoning the transmission of one "file" and starting the transmission of a new "file."

The transmitter must clear its compression history and respond with a Reset-Ack each time it receives a Reset-Request, because it cannot know if previous Reset-Acks reached the receiver. The receiver need not do anything to its history when it receives a Reset-Ack, because the transmitter will simply not refer to any prior history (’deflate’ is a sliding-window compressor).

When the link is busy, one decompression error is usually followed by several more before the Reset-Ack can be received. It is undesirable to transmit Reset-Requests more frequently than the round-trip-time of the link, because redundant Reset-Requests cause unnecessary compression dictionary clearing. The receiver MAY transmit an additional Reset-Request each time it receives a compressed or uncompressed packet until it finally receives a Reset-Ack, but the receiver ought not transmit another Reset-Request until the Reset-Ack for the previous one is late. The receiver MUST transmit enough Reset-Request packets to ensure that the transmitter receives at least one. For example, the receiver might choose to not transmit another Reset-Request until after one second (or, of course, a Reset-Ack has been received and decompression resumed).

Data Expansion

'Deflate', as used in this standard, expands incompressible data by approximately 14-18 bytes (8 bytes worst-case at the 'deflate' level, two further bytes for the 'deflate' end-of-block and the zero-length synchronization block header, two bytes of sequence number, and two bytes to account for adding the PPP Protocol Field to the transmitted data unit).

The BSD Compress draft proposal[7] describes an escape mechanism for incompressible data that trades off a layering violation for the irritating complications of variable and potentially unpredictable effective MRU lengths. That direct escape mechanism (and much of the text of its description) is used here as well.

If an incompressible data packet does not fit within the MRU of the link, the packet MUST be sent in its original form without CCP encapsulation; PPP packets with significant data expansion that do not exceed the MRU of the link SHOULD be sent in their original form without CCP encapsulation. In both of these cases, the transmitter must increment the sequence number, as future encapsulated packets will depend on the correct reception of some number of unencapsulated packets.

When a PPP packet is received with PPP Protocol numbers in the range 0x0000 to 0x3FFF, (except, of course, 0xFD and 0xFB) it is assumed that the packet would have caused expansion. The packet is locally added to the compression history. (Given the definition of the 'deflate' format, a convenient method of doing this is to locally "decompress" a stored-block header of the appropriate length, followed by the actual data block; or the data can simply be appended to the receiver's history, depending on implementation details.)

Sending incompressible packets in their native encapsulation avoids maximum transmission unit complications. If uncompressed packets could be larger than their native form, then it would be necessary for the upper layers of an implementation to treat the PPP link as if it had a smaller MTU, to ensure that compressed incompressible packets are never larger than the negotiated PPP MTU.

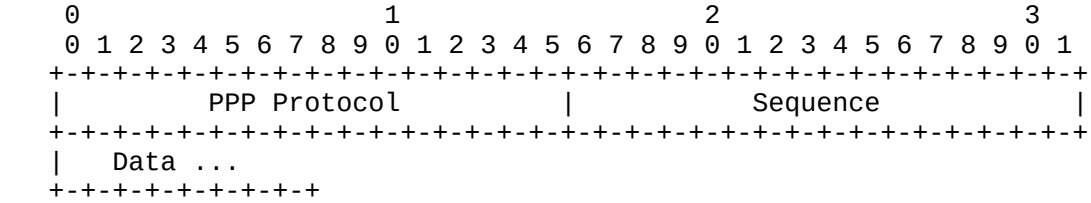
Using native encapsulation for incompressible packets complicates the implementation. The transmitter and the receiver must start putting information into the compression dictionary starting with the same packets, without relying upon seeing a compressed packet for synchronization. The first few packets after clearing the dictionary are usually incompressible, and so are likely to sent

in their native encapsulation, just like packets before compression is turned on. If CCP or LCP packets are handled separately from Network-Layer packets (e.g. a "daemon" for control packets and "kernel code" for data packets), care must be taken to ensure that the transmitter synchronizes clearing the dictionary with the transmission of the configure-ACK or Reset-Ack that starts compression, and the receiver must similarly ensure that its dictionary is cleared before it processes the next packet.

2.1. Packet Format

A summary of the PPP Deflate packet format is shown below.

The fields are transmitted from left to right.



PPP Protocol

The PPP Protocol field is described in the Point-to-Point Protocol Encapsulation [1].

When the PPP Deflate compression protocol is successfully negotiated by the PPP Compression Control Protocol [2], the value of the protocol field is 0xFD or 0xFB. This value MAY be compressed when Protocol-Field-Compression is negotiated.

Sequence

The sequence number is sent most significant octet first. It starts at 0 when the dictionary is cleared, and is incremented by 1 for each packet, including uncompressed packets. The sequence number after 65535 is zero. In other words, the sequence number "wraps" in the usual way.

The sequence number ensures that lost or out of order packets do not cause the compression databases of the peers to become unsynchronized. When an unexpected sequence number is encountered, the dictionaries must be resynchronized with a CCP Reset-Request or Configure-Request. The packet sequence number can be checked before a compressed packet is decoded.

Data

The compressed PPP encapsulated packet, consisting of the Protocol and Data fields of the original, uncompressed packet follows.

The Protocol field compression MUST be applied to the protocol field in the original packet before the sequence number is computed or the entire packet is compressed, regardless of whether the PPP protocol field compression has been negotiated. Thus, if the original protocol number was less than 0x100, it must be compressed to a single byte.

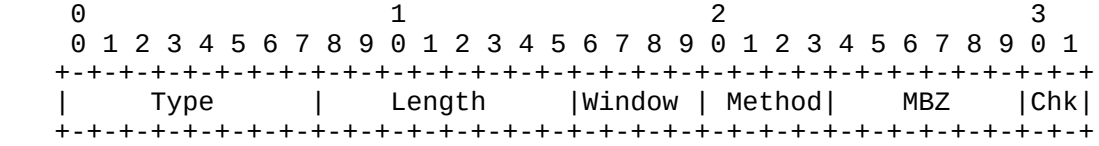
The basic format of the compressed data is precisely described by the 'Deflate' Compressed Data Format Specification[3]. Each transmitted packet must begin at a 'deflate' block boundary, to ensure synchronization when incompressible data resets the transmitter's state; to ensure this, each transmitted packet must be terminated with a zero-length 'deflate' non-compressed block (BTYPE of 00). This means that the last four bytes of the compressed format must be 0x00 0x00 0xFF 0xFF. These bytes MUST be removed before transmission; the receiver can reinsert them if required by the implementation.

3. Configuration Option Format

Description

The CCP PPP Deflate Configuration Option negotiates the use of PPP Deflate on the link. By default or ultimate disagreement, no compression is used.

A summary of the PPP Deflate Configuration Option format is shown below. The fields are transmitted from left to right.



Type

26 for PPP Deflate.

Length

3

Window

Represents the maximum window size the decompressor is willing to allocate; expressed as the base-2 logarithm of the LZ77 window size, minus 8. 'Deflate' compliant decompressors must be willing to accept the maximum 32KB window size, represented by a value of 7. A 'deflate' compliant compressor is at liberty to use a reduced window size, so a PPP Deflate compressor MUST either honor the restriction requested or reject the option.

Method

Must be the binary number 1000. Represents the 'zlib' Compression Method identifier of '8' for 'deflate' compression with up to 32K window size.

MBZ

Must be all 0 bits.

Chk

Must be 00 to specify sequence number check method.

Security Considerations

Security issues are not discussed in this memo.

References

[1] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[2] Rand, D., "The PPP Compression Control Protocol (CCP)", RFC 1962, June 1996.

[3] Deutsch, L.P., "'Deflate' Compressed Data Format Specification", draft available in ftp.uu.net:/pub/archiving/zip/doc/deflate-1.1.doc.

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

[5] 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.uu.net:/pub/archiving/zip/.

[6] Simpson, W., "PPP LCP Extensions", RFC 1570, January 1994.

[7] Schryver, V., "PPP BSD Compression Protocol", RFC 1977, August 1996.

Acknowledgments

William Simpson provided the very valuable idea of not using any additional header bytes for incompressible packets.

Chair's Address

The working group can be contacted via the current chair:

Karl Fox
Ascend Communications
3518 Riverside Drive, Suite 101
Columbus, Ohio 43221

EMail: karl@ascend.com

Author's Address

Questions about this memo can also be directed to:

John Woods
Proteon, Inc.
9 Technology Drive
Westborough MA 01581-1799

(508) 898-2800 ext. 2651
EMail: jfw@funhouse.com