Network Working Group
Internet Draft

Bellovin, Buchsbaum, Muthukrishnan
AT&T Labs--Research

Expiration Date: April 2000

October 1999

TCP Compression Filter

draft-bellovin-tcpcomp-00.txt

1. Status of this Memo

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 its 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."

This draft document will be submitted to the RFC Editor as an Experimental RFC. Distribution of this document is unlimited.

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.

2. Abstract

We propose a TCP filter option to install compression in a virtual layer between TCP and the application layer. The method is incrementally deployable, as neither party will install the compression layer without the other's consent.

3. Introduction

The natural place to compress data is at the application level, where application-specific semantics can be used to attain better compression. Unfortunately, this requires changing each and every application, or at least changing user behavior.

An alternative is to compress the data at the IP level, as is done in IPCOMP [RFC2393]. While this is application independent, its effectiveness is also limited, since each packet must be compressed individually.

We propose compressing immediately above TCP, as negotiated by a TCP option. One side sends an ordered list of which compression algorithms it supports. The other side selects one from the list, which commits both sides to compressing the payloads of all subsequent packets accordingly.

An example of where this could help is the transmission of email messages with large attachments, often word processor documents or slide presentations. Files of these types are quite compressible; doing the compression at a higher layer, however, would require either manual user intervention or changes to many different mail sending and receiving packages.

This option is an example of a TCP filter option of the class described in [FILTDRAFT].

3.1. Specification of Requirements

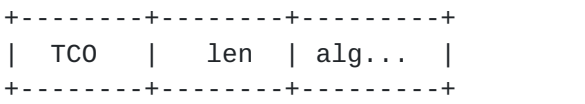
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC-2119].

4. Option Format

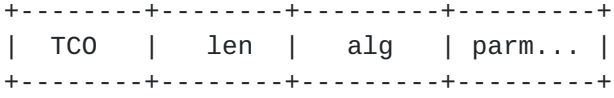
We use one TCP option, of type TCO (to be assigned by IANA), to signal compression. A type field indicates the operation.

A compression algorithm announcement MUST NOT appear except as specified by the [FILTDRAFT] protocol. (All TCPs MUST ignore unknown options in SYN packets [RFC1122].) Compressed packets MUST NOT be sent unless both parties have agreed to the appropriate filter via the protocol [FILTDRAFT].

Compression algorithm IDs will be assigned by IANA.



The compression filters, including any parameters, are fixed during the three-way handshake by the protocol [FILTDRAFT]. Subsequently, they may only be changed by in-band communication, i.e., by the compression algorithms themselves interpreting the data stream.



5. Behavior

As per [FILTDRAFT], by "initiator," we indicate the party that first includes compression options in its SYN packet, and by "respondent," we indicate the other party.

If the respondent (cf., [FILTDRAFT] protocol) has indicated that it can accept a compression algorithm, a sender MUST use it. As described in Section 2.2 of RFC 2393, the initiator SHOULD verify that compression does not increase the size of the message. If it does, it SHOULD NOT initiate compression.

Only the initial compression algorithms and parameters are determined by the compression options in the handshake. Senders MAY apply hysteresis to sending both compressed and uncompressed packets, per RFC 2393, but only by using in-band communication, i.e., messages in the data stream itself. In particular, to permit uncompressed data to be co-mingled with compressed data, we anticipate that particular compression algorithms will include their own header structures in the data stream. Again, note that because of the stream nature of TCP, the uncompressed portion may be sent in the same packet as

compressed data. Any necessary framing must be done by particular compression algorithms. They may, however, specify the use of the TCP record mark filter option [DRAFTREC].

Senders MUST honor the compression algorithm specified by the respondent, as per [FILTDRAFT]. Local dictates to the contrary require in-band communication to alter the compression behavior; if the compression algorithm precludes such communication, then the session must be terminated and re-established with different (or absent) compression options.

6. Interactions

6.1. TCP Urgent Pointer

Compression filters must note application requests to send urgent data. The urgent pointer passed down to TCP must point to the appropriate compressed bytes. Upon receipt of an urgent packet (more precisely, a packet where the urgent pointer denotes a byte within it), the uncompression routine must send the appropriate notification to the application, while pointing to the proper uncompressed byte.

7. Security Considerations

Compressing data above the TCP layer should not have any negative impact on security. In particular, port numbers are not compressed. Some firewalls and intrusion detection systems examine TCP payload data, however, and they may be confused by compression. The former may wish to delete the compression option; if the latter are used, administrators may wish to disable compression.

8. Acknowledgements

9. References

10. Appendix

Initial compression algorithms to be supported SHOULD include DEFLATE [RFC2394] (algorithm code 0x00) and LZS [RFC2395] (algorithm code 0x01).

11. Author Information

Steven M. Bellovin
smb@research.att.com
+1 973-360-8656

Adam L. Buchsbaum
alb@research.att.com
+1 973-360-8674

S. Muthukrishnan
muthu@research.att.com
+1 973-360-7212

AT&T Labs--Research
Shannon Laboratory
180 Park Avenue
Florham Park, NJ 07974