Status of this Memo

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

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.

Abstract

This document describes how to efficiently implement certain mechanisms of unidirectional operation in ROHC-TCP (robust header compression scheme for TCP/IP), RFC XXX, which can significantly improve the compression efficiency compared to using simple control scheme.

All the operational modes and state machines mentioned in this document are the same as the ones described in [ROHC-TCP].
Table of Contents

Status of this Memo................................................1
Abstract...........................................................1
1. Introduction....................................................4
2. Terminology....................................................5
3. Tracking-based TCP congestion window estimation...........6
   3.1. General principle of congestion window tracking.........6
   3.2. Tracking based on Sequence Number.....................6
   3.3. Tracking based on Acknowledgment Number...............8
   3.4. Further discussion on congestion window tracking.....9
4. Optional enhancements in Unidirectional mode...............10
   4.1. Optional operation for upwards transition.............10
      4.1.1. Optional transition for short-live TCP transfers....10
      4.1.2. Optional operation in IR state.....................10
      4.1.3. Optional operation in CO state.....................11
      4.1.4. Determine the value K in congestion window estimation11
   4.2. Optional operation for downwards transition...........12
   4.3. Other possible optimizations........................12
5. Security considerations......................................13
6. IANA Considerations........................................13
7. Acknowledgements...........................................14
8. Authors’ addresses...........................................14
9. Intellectual Property Right Considerations................14
10. References................................................14
Document History

00  November, 2002    First release.
01  May, 2003         Editorial revision.
1. Introduction

This document describes how to implement certain mechanisms in [ROHC-TCP] to significantly improve the compression efficiency in unidirectional operation mode comparing with the ones obtained with simple control scheme.

As discussed in [TCPBEH], Window-based LSB encoding [RFC3095], which does not assume the linear changing pattern of the target header fields, is more suitable to encode some TCP fields, such as SEQUENCE NUM, ACKNOWLEDGEMENT NUM, etc., both efficiently and robustly, considering the changing pattern of these TCP fields.

Using ROHC-TCP, both the compressor and decompressor maintain the context values. To provide robustness, the compressor can maintain more than one context value for each field. These values represent the r most likely candidates’ values for the context at the decompressor.

ROHC-TCP ensures that the compressed header contains enough information so that the uncompressed header can be extracted no matter which one of the compressor context values is actually stored at the decompressor. Storing more context values at the compressor reduces the chance that the decompressor will have a context value different from any of the values stored at the compressor (which could cause the packet to be decompressed incorrectly).

As an example, an implementation may choose to store the last r values of each field in the compressor context. In this case robustness is guaranteed against up to r - 1 consecutive dropped packets between the compressor and the decompressor. Thus, by keeping the value r large enough, the compressor rarely gets out of synchronization with the decompressor.

However, the trade-off is that the larger the number of context values is, the compressed header will be larger because it must contain more information to decompress relative to any of the candidate context values. That is to say, the compression efficiency will be reduced.

To reduce the number of context value r, the compressor needs some form of feedback to get sufficient confidence that a certain context value will not be used as a reference by the decompressor anymore. Then the compressor can remove that value and all other values older than it from its context. Obviously, when a feedback channel is available, this type of confidence can be achieved by proactive feedback in the form of ACKs from the decompressor. A feedback channel, however, is unavailable in unidirectional operational mode in ROHC-TCP. To simplify the description, the mechanism used previously in the ROHC-TCP compression process, is referred to as static control scheme.
One thing will be emphasized in this document is that, an implicit feedback can be obtained from the nature feedback-loop of TCP protocol itself for TCP/IP header compression even in unidirectional operational mode.

Since TCP is a window-based protocol, a new segment cannot be transmitted without getting the acknowledgment of segment in the previous window. Upon receiving the new segment, the compressor can get enough confidence that the decompressor has received the segment in the previous window and then can shrink the context window by removing all the values older than that segment.

This is to say, the context window of ROHC-TCP, the number of context value $r$, can be controlled by the TCP congestion window.

A tracking based TCP congestion window estimation algorithm is described in this document to estimate TCP congestion window in the compressor side.

All the operational modes and state machines mentioned in this document are the same as the ones described in [ROHC-TCP]. For better understanding of this draft, the reader should be familiar with the concept of [ROHC-TCP].

2. Terminology

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 [RFC2119].

TCP congestion window (cwnd)

A TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP MUST NOT send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd (RECEIVER WINDOW).

Loss propagation

Loss of headers, due to errors in (i.e., loss of or damage to) previous header(s) or feedback.

Robustness

The performance of a header compression scheme can be described with three parameters: compression efficiency, robustness, and compression transparency. A robust scheme tolerates loss and residual errors on the link over which header compression takes place without losing additional packets or introducing additional errors in decompressed headers.
3. Tracking-based TCP congestion window estimation

As originally outlined in [CAC] and specified in [RFC2581], TCP is incorporated with four congestion control algorithms: slow-start, congestion-avoidance, fast retransmit, and fast recovery. The effective window of TCP is mainly controlled by the congestion window and may change during the entire connection life. The enhancement of ROHC-TCP, ENH-ROHC-TCP, designs a mechanism to track the dynamics of TCP congestion window, and control the number of context value r of windowed LSB encoding by the estimated congestion window. By combining the windowed LSB encoding and TCP congestion window tracking, ENH-ROHC-TCP can achieve better performance over high bit-error-rate links.

Note that in one-way TCP traffic, only the information about sequence number or acknowledgment number is available for tracking TCP congestion window. ROHC-TCP does not require that all one-way TCP traffics must cross the same compressor. The detail will be described in the following sections.

3.1. General principle of congestion window tracking

The general principle of congestion window tracking is as follows. The compressor imitates the congestion control behavior of TCP upon receiving each segment, in the meantime, estimates the congestion window (cwnd) and the slow start threshold (ssthresh). Besides the requirement of accuracy, there are also some other requirements for the congestion window tracking algorithms:

- Simplex link. The tracking algorithm SHOULD always only take Sequence Number or Acknowledgment Number of a one-way TCP traffic into consideration. It SHOULD NOT use Sequence Number and Acknowledgment Number of that traffic simultaneously.

- Misordering resilience. The tracking algorithm SHOULD work well while receiving misordered segments.

- Multiple-links. The tracking algorithm SHOULD work well when not all the one-way TCP traffics are crossing the same link.

- Slightly overestimation. If the tracking algorithm cannot guarantee the accuracy of the estimated cwnd and ssthresh, it is RECOMMENDED that it produces a slightly overestimated one.

The following sections will describe two congestion window tracking algorithms, which use Sequence Number and Acknowledgment Number of a one-way TCP traffic, respectively.

3.2. Tracking based on Sequence Number
This algorithm (Algorithm SEQ) contains 3 states: SLOW-START, CONGESTION-AVOIDANCE, and FAST-RECOVERY, which are equivalent to the states in TCP congestion control algorithms. It maintains 2 variables: cwnd and ssthresh.

Initially, this algorithm starts in SLOW-START state with ssthresh set to ISSTHRESH and cwnd set to IW.

Upon receiving a segment, if it is the first segment, which is not necessary to be the SYN segment, the algorithm sets the current maximum Sequence Number (CMAXSN) and the current minimum Sequence Number (CMINSN) to this segment’s sequence number; otherwise, the algorithm takes a procedure according to the current state.

- SLOW-START
  * If the new Sequence Number (NSN) is larger than CMAXSN, increase cwnd by the distance between NSN and CMAXSN, and update CMAXSN and CMINSN based on the following rules:
    CMAXSN = NSN
    if (CMAXSN - CMINSN) > cwnd
    CMINSN = cwnd - CMAXSN;
  * If the cwnd is larger than ssthresh, the algorithm transits to CONGESTION-AVOIDANCE state;
  * If the distance between NSN and CMAXSN is less than or equal to 3*MSS, ignore it;
  * If the distance is larger than 3*MSS, halve the cwnd and set ssthresh to MAX(cwnd, 2*MSS). After that, the algorithm transits into FAST-RECOVERY state.

- CONGESTION-AVOIDANCE
  * If NSN is larger than CMAXSN, increase cwnd by ((NSN - CMAXSN)*MSS)/cwnd and then update CMAXSN and CMINSN based on...
the following rules:

\[
\text{CMAXSN} = \text{NSN}
\]

\[
\text{if (CMAXSN} - \text{CMINSN}) > \text{cwnd})
\]

\[
\text{CMINSN} = \text{cwnd} - \text{CMAXSN};
\]

* If the distance between NSN and CMAXSN is less than or equal to 3*MSS, ignore it;

* If the distance is larger than 3*MSS, halve the cwnd and set ssthresh to MAX(cwnd, 2*MSS). After that, the algorithm transits into FAST-RECOVERY state.

- FAST-RECOVERY

* If NSN is larger than or equal to CMAXSN + cwnd, the algorithm transits into CONGESTION-AVOIDANCE state;

* Otherwise, ignore it.

In this algorithm, MSS is denoted as the estimated maximum segment size. The implementation can use the MTU of the link as an approximation of this value. ISSTHRESH and IW are the initial values of ssthresh and cwnd, respectively. ISSTHRESH MAY be arbitrarily high. IW SHOULD be set to 4*MSS.

3.3. Tracking based on Acknowledgment Number

This algorithm (Algorithm ACK) maintains 3 states: SLOW-START, CONGESTION-AVOIDANCE and FAST-RECOVERY, which are equivalent to the states in TCP congestion control algorithms. It also maintains 2 variables: cwnd and ssthresh.

Initially, this algorithm starts in SLOW-START state with ssthresh set to ISSTHRESH and cwnd set to IW.

Upon receiving a segment, if it is the first segment, which is not necessary to be the SYN segment, the algorithm sets the current
maximum Acknowledgment Number (CMA\_ACK) to this segment’s acknowledgment number; otherwise, the algorithm takes a procedure according to the current state.

- SLOW-START

* If the new Acknowledgment Number (NEW\_ACK) is larger than CMA\_ACK, increase cwnd by the distance between NEW\_ACK and CMA\_ACK, set duplicate ack counter (NDUP\_ACKS) to 0, and update CMA\_ACK accordingly; If the cwnd is larger than ssthresh, the algorithm transits to CONGESTION-AVOIDANCE state;

* If NEW\_ACK is equal to CMA\_ACK, increase the NDUP\_ACKS by 1. If NDUP\_ACKS is greater than 3, halve the cwnd and set ssthresh to MAX(cwnd, 2*MSS). Consequently, the algorithm transits into FAST-RECOVERY state;

* Otherwise, set NDUP\_ACKS to 0.

- CONGESTION-AVOIDANCE

* If NEW\_ACK is larger than CMA\_ACK, increase cwnd by ((NEW\_ACK- CMA\_ACK)*MSS)/cwnd, set NDUP\_ACKS to 0 and update CMA\_ACK accordingly;

* If NEW\_ACK is equal to CMA\_ACK, increase NDUP\_ACKS by 1. If NDUP\_ACKS is greater than 3, halve the cwnd and set ssthresh to MAX(cwnd, 2*MSS). After that, the algorithm transits into FAST-RECOVERY state;

* Otherwise, set NDUP\_ACKS to 0.

- FAST-RECOVERY

* If NEW\_ACK is larger than CMA\_ACK, set NDUP\_ACKS to 0. Consequently, the algorithm transits into CONGESTION-AVOID state;

* Otherwise, ignore it.

In this algorithm, MSS is denoted as the estimated maximum segment size. The implementation can use the MTU of the link as an approximation of this value. ISS\_THRESH and IW are the initial values of ssthresh and cwnd, respectively. ISS\_THRESH MAY be arbitrarily high. IW SHOULD be set to 4*MSS.

### 3.4. Further discussion on congestion window tracking

In some cases, it is inevitable for the tracking algorithms to overestimate the TCP congestion window. Also, it SHOULD be avoided that the estimated congestion window gets significantly smaller than the actual one. For all of these cases, ROHC-TCP simply applies two
boundaries on the estimated congestion window size. One of the two
boundaries is the MIN boundary, which is the minimum congestion
window size and whose value is determined according to the [INITWIN];
the other boundary is the MAX boundary, which is the maximum
congestion window size. There are two possible approaches to
setting this MAX boundary. One is to select a commonly used maximum
TCP socket buffer size. The other one is to use the simple equation
\[ W = \sqrt{\frac{8}{3}l} \], where \( W \) is the maximum window size and \( l \) is the
typical packet loss rate.

If ECN mechanism is deployed, according to [RFC2481] and [ECN], the
TCP sender will set the CWR (Congestion Window Reduced) flag in the
TCP header of the first new data packet sent after the window
reduction, and the TCP receiver will reset the ECN-Echo flag back to
0 after receiving a packet with CWR flag set. Thus, the CWR flag
and the ECN-Echo flag’s transition from 1 to 0 can be used as
another indication of congestion combined with other mechanisms
mentioned in the tracking algorithm.

4. Optional enhancements in Unidirectional mode

Several implementation enhancements will be introduced in this
section to improve the compression efficiency in unidirectional mode
by utilizing the TCP congestion window estimated using the above
mechanism.

4.1. Optional operations for upwards transition

4.1.1. Optional transition for short-live TCP transfers

This approach is introduced in ENH-ROHC-TCP to compress short-lived
TCP transfers more efficiently.

The key message of this approach is that the compressor should try
to speed up the initialization process. This approach can be
applied if the compressor is able to see the SYN packet. The
compressor enters the IR state when it receives the packet with SYN
bit set and sends IR packet. When it receives the first data packet
of the transfer, it should transit to FO state because that means
the decompressor has received the packet with SYN bit set and
established the context successfully at its side. Using this
mechanism can significantly reduce the number of context initiation
headers.

4.1.2. Optional operation in IR state

In IR state, the compressor can send full header (or partial full
header) periodically with an exponentially increasing period, which
is so-called compression slow-start [RFC2507].

The main idea of this optional operation is controlling the size of
context window by dynamically TCP congestion window estimation.
After a packet has been sent out, the compressor invokes the Algorithm SEQ or Algorithm ACK introduced in the above session to track the congestion windows of the two one-way traffics with different directions in a TCP connection. Suppose that the estimated congestion windows are $cwnd_{seq}$ and $cwnd_{ack}$, while the estimated slow start thresholds are $ssthresh_{seq}$ and $ssthresh_{ack}$, respectively. Let

$$W(cwnd_{seq}, ssthresh_{seq}, cwnd_{ack}, ssthresh_{ack}) = K \cdot \max(\max(cwnd_{seq}, 2ssthresh_{seq}), \max(cwnd_{ack}, 2ssthresh_{ack})).$$

If the value of context window, $r$, is larger than $W(cwnd_{seq}, ssthresh_{seq}, cwnd_{ack}, ssthresh_{ack})$, $r$ can be reduced. $K$ is an implementation parameter that will be further discussed in Section 4.1.4.

If the number of the compress packets been send gets larger than $W(cwnd_{seq}, ssthresh_{seq}, cwnd_{ack}, ssthresh_{ack})$, the compressor transits to CO state.

If the compressor transits to the IR state from the higher states, the compressor will re-initialize the algorithm for tracking TCP congestion window.

4.1.3. Optional operation in CO state

After a packet has been sent out, the compressor invokes the Algorithm SEQ or Algorithm ACK to track the congestion windows of the two one-way traffics with different directions in a TCP connection. Suppose that the estimated congestion windows are $cwnd_{seq}$ and $cwnd_{ack}$, while the estimated slow start thresholds are $ssthresh_{seq}$ and $ssthresh_{ack}$, respectively. Let

$$W(cwnd_{seq}, ssthresh_{seq}, cwnd_{ack}, ssthresh_{ack}) = K \cdot \max(\max(cwnd_{seq}, 2ssthresh_{seq}), \max(cwnd_{ack}, 2ssthresh_{ack})).$$

If the value of context window, $r$, is larger than $W(cwnd_{seq}, ssthresh_{seq}, cwnd_{ack}, ssthresh_{ack})$, $r$ can be reduced. $K$ is an implementation parameter, which can be set to the same value as in the IR state.

If the compressor finds that the payload size of consecutive packets is a constant value and one of such packets has been removed from the context window, which means the decompressor has known the exact value of the constant size, it may use fixed-payload encoding scheme to improve the compression efficiency.

4.1.4. Determine the value $K$ in congestion window estimation
As mentioned above, the context window SHOULD be shrunk when its size gets larger than $K \times \max(\max(cwnd_{seq}, 2ssthresh_{seq}), \max(cwnd_{ack}, 2ssthresh_{ack}))$. Since the Fast Recovery algorithm was introduced in TCP, several TCP variants had been proposed, which are different only in the behaviors of Fast Recovery. Some of them need several RTTs to be recovered from multiple losses in a window. Ideally, they may send $L \times W/2$ packets in this stage, where $L$ is the number of lost packets and $W$ is the size of the congestion window where error occurs. Some recent work [REQTCP] on improving TCP performance allows transmitting packets even when receiving duplicate acknowledgments. Due to the above concerns, it’d better keep $K$ large enough so as to prevent shrinking context window without enough confidence that corresponding packets had been successfully received.

Considering the bandwidth-limited environments and the limited receiver buffer, a practical range of $K$ is around 1-2. From the simulation results, $K=1$ is good enough for most cases.

4.2. Optional operation for downwards transition

The compressor must immediately transit back to the IR state when the header to be compressed, falls behind the context window, i.e. it is older than all the packets in context.

If the context window contains only one packet, which means there is a long jump in the packet sequence number or acknowledge number, the compressor will transit to the IR state. Here, a segment causes a long jump when the distance between its sequence number (or acknowledgment number) and $CMAXSN$ (or $CMAXACK$) is larger than the estimated congestion window size, i.e.,

$$|\text{sequence number (acknowledgement number)} - CMAXSN (CMAXACK)| > \text{estimated congestion window size}.$$

4.3. Other possible optimizations

It can be seen that there are two distinct deployments - one where the forward and reverse paths share a link and one where they do not.

In the former case it may be the situation that a compressor and decompressor co-locate as illustrated in Figure 4.3. It may then be possible for the compressor and decompressor at each end of the link to exchange information. In such a scenario, ENH-ROHC-TCP can make further optimization on context size for windowed LSB encoding.

In Figure 4.3., C-SN represents the compressor for the sequence number traffic that deployed in Host A, D-SN represents the decompressor for the sequence number traffic that deployed in Host B. Similarly, C-ACK represents the compressor for the acknowledgement number traffic that deployed in Host B, D-ACK represents the
decompressor for the acknowledgement number traffic that deployed in Host A.

![Figure 4.3. Illustration of Possible optimization in U-mode.](image)

It is known that acknowledgement numbers (from C-ACK to D-ACK) are generally taken from the sequence numbers (from C-SN to D-SN) in the opposite direction. Since an acknowledgement cannot be generated for a packet that has not passed across the link, this offers an efficient way of estimating the TCP congestion window.

Denote the current sequence number that is sending out from C-SN as SN-New, denote the sequence number that been acknowledged by the D-ACK as SN-Old, then the TCP congestion window (cwnd-bidir) can be represented as

\[
cwnd-bidir = SN-New - SN-Old.
\]

Having this new estimated congestion window, the control parameter \( W \) will be re-calculated as

\[
W(cwnd_seq, ssthresh_seq, cwnd_ack, ssthresh_ack) = \min(W(cwnd_seq, ssthresh_seq, cwnd_ack, ssthresh_ack), cwnd-bidir)
\]

5. Security considerations

ROHC-TCP is conformed to ROHC framework. Consequently the security considerations for this enhancement document match those of [ROHC-TCP].

6. IANA Considerations

This document does not require any IANA involvement.
7. Acknowledgements

Thanks to Richard Price, Mark A West, Mikael Degermark, and Carsten Bormann for valuable input.

8. Authors’ addresses

Qian Zhang              Tel: +86 10 62617711-3135
Email:                  qianz@microsoft.com

HongBin Liao           Tel: +86 10 62617711-3156
Email:                  i-hbliao@microsoft.com

Wenwu Zhu              Tel: +86 10 62617711-5405
Email:                  wwzhu@microsoft.com

Ya-Qin Zhang           Tel: +86 10 62617711
Email:                  yzhang@microsoft.com

Microsoft Research Asia
Beijing Sigma Center
No.49, Zhichun Road, Haidian District
Beijing 100080, P.R.C.

9. Intellectual Property Right Considerations

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

10. References


M. Allman, S. Floyd, and C. Partridge, "Increasing TCP’s Initial Window", Internet Draft (work in progress), May 2001. <draft-ietf-tsvwg-initwin-00.txt>


L-E. Jonsson, "Requirements for ROHC IP/TCP header compression", Internet Draft (work in progress), Nov. 2002. <draft-ietf-rohc-tcp-requirements-05.txt>