Internet Engineering Task Force (IETF)
Request for Comments: 6069
Category: Experimental
ISSN: 2070-1721
December 2010
A. Zimmermann
A. Hannemann
RWTH Aachen University

Making TCP More Robust to Long Connectivity Disruptions (TCP-LCD)

Abstract

Disruptions in end-to-end path connectivity, which last longer than one retransmission timeout, cause suboptimal TCP performance. The reason for this performance degradation is that TCP interprets segment loss induced by long connectivity disruptions as a sign of congestion, resulting in repeated retransmission timer backoffs. This, in turn, leads to a delayed detection of the re-establishment of the connection since TCP waits for the next retransmission timeout before it attempts a retransmission.

This document proposes an algorithm to make TCP more robust to long connectivity disruptions (TCP-LCD). It describes how standard ICMP messages can be exploited during timeout-based loss recovery to disambiguate true congestion loss from non-congestion loss caused by connectivity disruptions. Moreover, a reversion strategy of the retransmission timer is specified that enables a more prompt detection of whether or not the connectivity to a previously disconnected peer node has been restored. TCP-LCD is a TCP sender-only modification that effectively improves TCP performance in the case of connectivity disruptions.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Connectivity disruptions can occur in many different situations. The frequency of connectivity disruptions depends on the properties of the end-to-end path between the communicating hosts. While connectivity disruptions can occur in traditional wired networks, e.g., disruption caused by an unplugged network cable, the likelihood of their occurrence is significantly higher in wireless (multi-hop) networks. Especially, end-host mobility, network topology changes, and wireless interferences are crucial factors. In the case of the Transmission Control Protocol (TCP) [RFC0793], the performance of the connection can experience a significant reduction compared to a permanently connected path [SESB05]. This is because TCP, which was originally designed to operate in fixed and wired networks, generally assumes that the end-to-end path connectivity is relatively stable over the connection's lifetime.

Depending on their duration, connectivity disruptions can be classified into two groups [TCP-RLCI]: "short" and "long". A connectivity disruption is "short" if connectivity returns before the retransmission timer fires for the first time. In this case, TCP recovers lost data segments through Fast Retransmit and lost acknowledgments (ACKs) through successfully delivered later ACKs. Connectivity disruptions are declared as "long" for a given TCP connection if the retransmission timer fires at least once before connectivity is resumed. Whether or not path characteristics, like the round-trip time (RTT) or the available bandwidth, have changed when connectivity resumes after a disruption is another important aspect for TCP's retransmission scheme [TCP-RLCI].

The algorithm specified in this document improves TCP's behavior in the case of "long connectivity disruptions". In particular, it focuses on the period prior to the re-establishment of the connectivity to a previously disconnected peer node. The document does not describe any modifications to TCP's behavior and its congestion control mechanisms [RFC5681] after connectivity has been restored.

When a long connectivity disruption occurs on a TCP connection, the TCP sender eventually does not receive any more acknowledgments. After the retransmission timer expires, the TCP sender enters the timeout-based loss recovery and declares the oldest outstanding segment (SND.UNA) as lost. Since TCP tightly couples reliability and congestion control, the retransmission of SND.UNA is triggered together with the reduction of the transmission rate. This is based on the assumption that segment loss is an indication of congestion [RFC5681]. As long as the connectivity disruption persists, TCP will repeat this procedure until the oldest outstanding segment has successfully been acknowledged or until the connection has timed out. TCP implementations that follow the recommended retransmission timeout (RTO) management of RFC 2988 [RFC2988] double the RTO after each retransmission attempt. However, the RTO growth may be bounded by an upper limit, the maximum RTO, which is at least 60 s, but may be longer: Linux, for example, uses 120 s. If connectivity is restored between two retransmission attempts, TCP still has to wait until the retransmission timer expires before resuming transmission, since it simply does not have any means to know if the connectivity has been re-established. Therefore, depending on when connectivity becomes available again, this can waste up to a maximum RTO of possible transmission time.

This retransmission behavior is not efficient, especially in scenarios with long connectivity disruptions. In the ideal case, TCP would attempt a retransmission as soon as connectivity to its peer has been re-established. In this document, we specify a TCP sender-only modification to provide robustness to long connectivity disruptions (TCP-LCD). The memo describes how the standard Internet Control Message Protocol (ICMP) can be exploited during timeout-based loss recovery to identify non-congestion loss caused by long connectivity disruptions. TCP-LCD's reversion strategy of the retransmission timer enables higher-frequency retransmissions and thereby a prompt detection when connectivity to a previously disconnected peer node has been restored. If no congestion is present, TCP-LCD approaches the ideal behavior.

Experimental results of a Linux implementation of TCP-LCD have been presented in [ZimHan09]. The implementation has been incorporated into mainline Linux, and is already used within the Internet. Thus far, no negative experiences have been reported that could be attributed to the algorithm. However, we consider TCP-LCD as experimental until more real-life results have been obtained. Nevertheless, we encourage implementation of TCP-LCD under other operating systems to provide for broader testing and experimentation opportunities.

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 [RFC2119].

The reader should be familiar with the algorithm and terminology from [RFC2988], which defines the standard algorithm that Transmission Control Protocol (TCP) senders are required to use to compute and manage their retransmission timer. In this document, the terms "retransmission timer" and "retransmission timeout" are used as defined in [RFC2988]. The retransmission timer ensures data delivery in the absence of any feedback from the receiver. The duration of this timer is referred to as retransmission timeout (RTO).

As defined in [RFC0793], the term "acceptable acknowledgment (ACK)" refers to a TCP segment that acknowledges previously unacknowledged data. The TCP sender state variable "SND.UNA" and the current segment variable "SEG.SEQ" are used as defined in [RFC0793]. SND.UNA holds the segment sequence number of the earliest segment that has not been acknowledged by the TCP receiver (the oldest outstanding segment). SEG.SEQ is the segment sequence number of a given segment.

For the purposes of this specification, we define the term "timeout-based loss recovery", which refers to the state that a TCP sender enters upon the first timeout of the oldest outstanding segment (SND.UNA) and leaves upon the arrival of the *first* acceptable ACK. It is important to note that other documents use a different interpretation of the term "timeout-based loss recovery". For example, the NewReno modification to TCP's Fast Recovery algorithm [RFC3782] extends the period that a TCP sender remains in timeout-based loss recovery compared to the one defined in this document. This is because [RFC3782] attempts to avoid unnecessary multiple Fast Retransmits that can occur after an RTO.

3. Connectivity Disruption Indication

If the queue of an intermediate router that is experiencing a link outage can buffer all incoming packets, a connectivity disruption will only cause a variation in delay, which is handled well by TCP implementations using either Eifel [RFC3522], [RFC4015] or Forward RTO-Recovery (F-RTO) [RFC5682]. However, if the link outage lasts for too long, the router experiencing the link outage is forced to drop packets, and finally may remove the corresponding next hop from its routing table. Means to detect such link outages include reacting to failed address resolution protocol (ARP) [RFC0826] queries, sensing unsuccessful links, and the like. However, this is solely the responsibility of the respective router.

> Note: The focus of this memo is on introducing a method of how ICMP messages may be exploited to improve TCP's performance; how different physical and link-layer mechanisms below the network layer may trigger ICMP destination unreachable messages are out of scope of this memo.

Provided that no other route to the specific destination exists, an Internet Protocol version 4 (IPv4) [RFC0791] router will notify the corresponding sending host about the dropped packets via ICMP destination unreachable messages of code 0 (net unreachable) or code 1 (host unreachable) [RFC1812]. Therefore, the sending host can use the ICMP destination unreachable messages of these codes as an indication of a connectivity disruption, since the reception of these messages provides evidence that packets were dropped due to a link outage.

For Internet Protocol version 6 (IPv6) [RFC2460], the counterpart of the ICMP destination unreachable message of code 0 (net unreachable) and of code 1 (host unreachable) is the ICMPv6 destination unreachable message of code 0 (no route to destination) [RFC4443]. As with IPv4, a router should generate an ICMPv6 destination unreachable message of code 0 in response to a packet that cannot be delivered to its destination address because it lacks a matching entry in its routing table.

Note that there are also other ICMP and ICMPv6 destination unreachable messages with different codes. Some of them are candidates for connectivity disruption indications, too, but need further investigation (for example, ICMP destination unreachable messages with code 5 (source route failed), code 11 (net unreachable for TOS (Type of Service)), or code 12 (host unreachable for TOS) [RFC1812]). On the other hand, codes that flag hard errors are of no use for this scheme, since TCP should abort the connection when those are received [RFC1122].

For the sake of simplicity, we will use, unless explicitly qualified with ICMPv4 or ICMPv6, the term "ICMP unreachable message" as a synonym for ICMP destination unreachable messages of code 0 or code 1 and ICMPv6 destination unreachable messages of code 0. This implies that all keywords from [RFC2119] that deal with the handling of received ICMP messages apply in the same way to ICMPv6 messages.

The accurate interpretation of ICMP unreachable messages as a connectivity disruption indication is complicated by the following two peculiarities of ICMP messages. First, they do not necessarily operate on the same timescale as the packets, i.e., TCP segments that elicited them. When a router drops a packet due to a missing route, it will not necessarily send an ICMP unreachable message immediately, but will rather queue it for later delivery. Second, ICMP messages are subject to rate-limiting, e.g., when a router drops a whole window of data due to a link outage, it is unlikely to send as many ICMP unreachable messages as dropped TCP segments. Depending on the load of the router, it may not even send any ICMP unreachable messages at all. Both peculiarities originate from [RFC1812] for ICMPv4 and [RFC4443] for ICMPv6.

Fortunately, according to [RFC0792], ICMPv4 unreachable messages have to contain, in their body, the entire IPv4 header [RFC0791] of the datagram eliciting the ICMPv4 unreachable message, plus the first 64 bits of the payload of that datagram. This allows the sending host to match the ICMPv4 error message to the transport connection that elicited it. RFC 1812 [RFC1812] augments these requirements and states that ICMPv4 messages should contain as much of the original datagram as possible without the length of the ICMPv4 datagram exceeding 576 bytes. Therefore, in the case of TCP, at least the source port number, the destination port number, and the 32-bit TCP sequence number are included. This allows the originating TCP to demultiplex the received ICMPv4 message and to identify the affected connection. Moreover, it can identify which segment of the respective connection triggered the ICMPv4 unreachable message, unless there are several segments in flight with the same sequence number (see Section 5.1).

For IPv6 [RFC2460], the payload of an ICMPv6 error message has to include as many bytes as possible from the IPv6 datagram that elicited the ICMPv6 error message, without making the error message exceed the minimum IPv6 MTU (1280 bytes) [RFC4443]. Thus, enough information is available to identify both the affected connection and the corresponding segment that triggered the ICMPv6 error message.

A connectivity disruption indication in the form of an ICMP unreachable message associated with a presumably lost TCP segment provides strong evidence that the segment was not dropped due to congestion, but was successfully delivered as far as the reporting router. It therefore did not witness any congestion at least on that part of the path that was traversed by both the TCP segment eliciting the ICMP unreachable message and the ICMP unreachable message itself.

4. Connectivity Disruption Reaction

Section 4.1 introduces the basic idea of TCP-LCD. The complete algorithm is specified in Section 4.2.

4.1. Basic Idea

The goal of the algorithm is to promptly detect when connectivity to a previously disconnected peer node has been restored after a long connectivity disruption, while retaining appropriate behavior in case of congestion. TCP-LCD exploits standard ICMP unreachable messages during timeout-based loss recovery. This increases TCP's retransmission frequency by undoing one retransmission timer backoff whenever an ICMP unreachable message is received that contains a segment with a sequence number of a presumably lost retransmission.

This approach has the advantage of appropriately reducing the probing rate in case of congestion. If either the retransmission itself or the corresponding ICMP message is dropped, the previously performed retransmission timer backoff is not undone, which effectively halves the probing rate.

4.2. Algorithm Details

A TCP sender that uses RFC 2988 [RFC2988] to compute TCP's retransmission timer MAY employ the following scheme to avoid over-conservative retransmission timer backoffs in case of long connectivity disruptions. If a TCP sender does implement the following steps, the algorithm MUST be initiated upon the first timeout of the oldest outstanding segment (SND.UNA) and MUST be stopped upon the arrival of the first acceptable ACK. The algorithm MUST NOT be re-initiated upon subsequent timeouts for the same segment. The scheme SHOULD NOT be used in SYN-SENT or SYN-RECEIVED states [RFC0793] (see Section 5.5).

A TCP sender that does not employ RFC 2988 [RFC2988] to compute TCP's retransmission timer MUST NOT use TCP-LCD. We envision that the scheme could be easily adapted to algorithms other than RFC 2988. However, we leave this as future work.

RFC 2988 [RFC2988] provides in rule (2.5) the option to place a maximum value on the RTO. When a TCP implements this rule to provide an upper bound for the RTO, it MUST also be used in the following algorithm. In particular, if the RTO is bounded by an upper limit (maximum RTO), the "MAX_RTO" variable used in this scheme MUST be initialized with this upper limit. Otherwise, if the RTO is unbounded, the "MAX_RTO" variable MUST be set to infinity.

The scheme specified in this document uses the "BACKOFF_CNT" variable, whose initial value is zero. The variable is used to count the number of performed retransmission timer backoffs during one timeout-based loss recovery. Moreover, the "RTO_BASE" variable is used to recover the previous RTO if the retransmission timer backoff was unnecessary. The variable is initialized with the RTO upon initiation of timeout-based loss recovery.

(1) Before TCP updates the variable "RTO" when it initiates timeout-based loss recovery, set the variables "BACKOFF_CNT" and "RTO_BASE" as follows:

```
BACKOFF_CNT := 0;
RTO_BASE := RTO.
```

Proceed to step (R).

(R) This is a placeholder for standard TCP's behavior in case the retransmission timer has expired. In particular, if RFC 2988 [RFC2988] is used, steps (5.4) to (5.6) of that algorithm go here. Proceed to step (2).

(2) To account for the expiration of the retransmission timer in the previous step (R), increment the "BACKOFF_CNT" variable by one:

    BACKOFF_CNT := BACKOFF_CNT + 1.

(3) Wait either

a) for the expiration of the retransmission timer. When the retransmission timer expires, proceed to step (R); or

b) for the arrival of an acceptable ACK. When an acceptable ACK arrives, proceed to step (A); or

c) for the arrival of an ICMP unreachable message. When the ICMP unreachable message "ICMP_DU" arrives, proceed to step (4).

(4) If "BACKOFF_CNT > 0", i.e., if at least one retransmission timer backoff can be undone, then

    proceed to step (5);

else

    proceed to step (3).

(5) Extract the TCP segment header included in the ICMP unreachable message "ICMP_DU":

    SEG := Extract(ICMP_DU).

(6) If "SEG.SEQ == SND.UNA", i.e., if the TCP segment "SEG" eliciting the ICMP unreachable message "ICMP_DU" contains the sequence number of a retransmission, then

    proceed to step (7);

else

    proceed to step (3).

(7) Undo the last retransmission timer backoff:

```
BACKOFF_CNT := BACKOFF_CNT - 1;
RTO := min(RTO_BASE * 2^(BACKOFF_CNT), MAX_RTO).
```

(8) If the retransmission timer expires due to the undoing in the previous step (7), then

```
   proceed to step (R);

else

   proceed to step (3).
```

(A) This is a placeholder for standard TCP's behavior in case an acceptable ACK has arrived. No further processing.

When a TCP in steady-state detects a segment loss using the retransmission timer, it enters the timeout-based loss recovery and initiates the algorithm (step (1)). It adjusts the slow-start threshold (ssthresh), sets the congestion window (cwnd) to one segment, backs off the retransmission timer, and retransmits the first unacknowledged segment (step (R)) [RFC5681], [RFC2988]. To account for the expiration of the retransmission timer, the TCP sender increments the "BACKOFF_CNT" variable by one (step (2)).

In case the retransmission timer expires again (step (3a)), a TCP will repeat the retransmission of the first unacknowledged segment and back off the retransmission timer once more (step (R)) [RFC2988], as well as increment the "BACKOFF_CNT" variable by one (step (2)). Note that a TCP may implement RFC 2988's [RFC2988] option to place a maximum value on the RTO that may result in not performing the retransmission timer backoff. However, step (2) MUST always and unconditionally be applied, no matter whether or not the retransmission timer is actually backed off. In other words, each time the retransmission timer expires, the "BACKOFF_CNT" variable MUST be incremented by one.

If the first received packet after the retransmission(s) is an acceptable ACK (step (3b)), a TCP will proceed as normal, i.e., slow-start the connection and terminate the algorithm (step (A)). Later ICMP unreachable messages from the just terminated timeout-based loss recovery are ignored, since the ACK clock is already restarting due to the successful retransmission.

On the other hand, if the first received packet after the retransmission(s) is an ICMP unreachable message (step (3c)), and if step (4) permits it, TCP SHOULD undo one backoff for each ICMP unreachable message reporting an error on a retransmission. To decide if an ICMP unreachable message was elicited by a retransmission, the sequence number it contains is inspected (step (5), step (6)). The undo is performed by recalculating the RTO with the decremented "BACKOFF_CNT" variable (step (7)). This calculation explicitly matches the (bounded) exponential backoff specified in rule (5.5) of [RFC2988].

Upon receipt of an ICMP unreachable message that legitimately undoes one backoff, there is the possibility that the shortened retransmission timer has already expired (step (8)). Then, TCP SHOULD retransmit immediately. In case the shortened retransmission timer has not yet expired, TCP MUST wait accordingly.

5. Discussion of TCP-LCD

TCP-LCD takes caution to only react to connectivity disruption indications in the form of ICMP unreachable messages during timeout-based loss recovery. Therefore, TCP's behavior is not altered when either no ICMP unreachable messages are received or the retransmission timer of the TCP sender did not expire since the last received acceptable ACK. Thus, by definition, the algorithm triggers only in the case of long connectivity disruptions.

Only such ICMP unreachable messages that contain a TCP segment with the sequence number of a retransmission, i.e., that contain SND.UNA, are evaluated by TCP-LCD. All other ICMP unreachable messages are ignored. The arrival of those ICMP unreachable messages provides strong evidence that the retransmissions were not dropped due to congestion, but were successfully delivered to the reporting router. In other words, there is no evidence for any congestion at least on that very part of the path that was traversed by both the TCP segment eliciting the ICMP unreachable message and the ICMP unreachable message itself.

However, there are some situations where TCP-LCD makes a false decision and incorrectly undoes a retransmission timer backoff. This can happen, even when the received ICMP unreachable message contains the segment number of a retransmission (SND.UNA), because the TCP segment that elicited the ICMP unreachable message may either not be a retransmission (Section 5.1) or does not belong to the current timeout-based loss recovery (Section 5.2). Finally, packet duplication (Section 5.3) can also spuriously trigger the algorithm.

Section 5.4 discusses possible probing frequencies, while Section 5.6 describes the motivation for not reacting to ICMP unreachable messages while TCP is in steady-state.

5.1. Retransmission Ambiguity

Historically, the retransmission ambiguity problem [Zh86], [KP87] is the TCP sender's inability to distinguish whether the first acceptable ACK after a retransmission refers to the original transmission or to the retransmission. This problem occurs after both a Fast Retransmit and a timeout-based retransmit. However, modern TCP implementations can eliminate the retransmission ambiguity with either the help of Eifel [RFC3522], [RFC4015] or Forward RTO-Recovery (F-RTO) [RFC5682].

The reversion strategy of the given algorithm suffers from a form of retransmission ambiguity, too. In contrast to the above case, TCP suffers from ambiguity regarding ICMP unreachable messages received during timeout-based loss recovery. With the TCP segment number included in the ICMP unreachable message, a TCP sender is not able to determine if the ICMP unreachable message refers to the original transmission or to any of the timeout-based retransmissions. That is, there is an ambiguity with regard to which TCP segment an ICMP unreachable message reports on.

However, this ambiguity is not considered to be a problem for the algorithm. The assumption that a received ICMP unreachable message provides evidence that a non-congestion loss caused by the connectivity disruption was wrongly considered a congestion loss still holds, regardless of to which TCP segment (transmission or retransmission) the message refers.

5.2. Wrapped Sequence Numbers

Besides the ambiguity whether a received ICMP unreachable message refers to the original transmission or to any of the retransmissions, there is another source of ambiguity related to the TCP sequence numbers contained in ICMP unreachable messages. For high-bandwidth paths, the sequence space may wrap quickly. This might cause delayed ICMP unreachable messages to coincidentally fit as valid input in the proposed scheme. As a result, the scheme may incorrectly undo retransmission timer backoffs. The chances of this happening are minuscule, since a particular ICMP unreachable message would need to contain the exact sequence number of the current oldest outstanding segment (SND.UNA), while at the same time TCP is in timeout-based loss recovery. However, two "worst case" scenarios for the algorithm are possible.

For instance, consider a steady-state TCP connection, which will be disrupted at an intermediate router due to a link outage. Upon the expiration of the RTO, the TCP sender enters the timeout-based loss recovery and starts to retransmit the earliest segment that has not been acknowledged (SND.UNA). For some reason, the router delays all corresponding ICMP unreachable messages so that the TCP sender backs the retransmission timer off normally without any undoing. At the end of the connectivity disruption, the TCP sender eventually detects the re-establishment, and it leaves the scheme and finally the timeout-based loss recovery, too. A sequence number wrap-around later, the connectivity between the two peers is disrupted again, but this time due to congestion and exactly at the time at which the current SND.UNA matches the SND.UNA from the previous cycle. If the router emits the delayed ICMP unreachable messages now, the TCP sender would incorrectly undo retransmission timer backoffs. As the TCP sequence number contains 32 bits, the probability of this scenario is at most 1/2^32. Given sufficiently many retransmissions in the first timeout-based loss recovery, the corresponding ICMP unreachable messages could reduce the RTO in the second recovery at most to "RTO_BASE". However, once the ICMP unreachable messages are depleted, the standard exponential backoff will be performed. Thus, the congestion response will only be delayed by some false retransmissions.

Similar to the above, consider the case where a steady-state TCP connection with n segments in flight will be disrupted at some point due to a link outage at an intermediate router. For each segment in flight, the router may generate an ICMP unreachable message. However, for some reason, it delays them. Once the link outage is over and the connection has been re-established, the TCP sender leaves the scheme and slow-starts the connection. Following a sequence number wrap-around, a retransmission timeout occurs, just at the moment the TCP sender's current window of data reaches the previous range of the sequence number space again. In case the router emits the delayed ICMP unreachable messages now, spurious undoing of the retransmission timer backoff is possible once, if the TCP segment number contained in the ICMP unreachable messages matches the current SND.UNA, and the timeout was a result of congestion. In the case of another connectivity disruption, the additional undoing of the retransmission timer backoff has no impact. The probability of this scenario is at most n/2^32.

5.3. Packet Duplication

In case an intermediate router duplicates packets, a TCP sender may receive more ICMP unreachable messages during timeout-based loss recovery than sent timeout-based retransmissions. However, since TCP-LCD keeps track of the number of performed retransmission timer backoffs in the "BACKOFF_CNT" variable, it will not undo more retransmission timer backoffs than were actually performed. Nevertheless, if packet duplication and congestion coincide on the path between the two communicating hosts, duplicated ICMP unreachable messages could hide the congestion loss of some retransmissions or ICMP unreachable messages, and the algorithm may incorrectly undo retransmission timer backoffs. Considering the overall impact of a router that duplicates packets, the additional load induced by some spurious timeout-based retransmits can probably be neglected.

5.4. Probing Frequency

One might argue that if an ICMP unreachable message arrives for a timeout-based retransmission, the RTO shall be reset or recalculated, similar to what is done when an ACK arrives during timeout-based loss recovery (see Karn's algorithm [KP87], [RFC2988]), and a new retransmission should be sent immediately. Generally, this would result in a much higher probing frequency based on the round-trip time to the router where connectivity has been disrupted. However, we believe the current scheme provides a good trade-off between conservative behavior and fast detection of connectivity re-establishment. TCP-LCD focuses on long-connectivity disruptions, i.e., on disruptions that last for several RTOs. Thus, a much higher probing frequency (less than once per RTO) would not significantly increase the available transmission time compared to the duration of the connectivity disruption.

5.5. Reaction during Connection Establishment

It is possible that a TCP sender enters timeout-based loss recovery while the connection is in SYN-SENT or SYN-RECEIVED states [RFC0793]. The algorithm described in this document could also be used for faster connection establishment in networks with connectivity disruptions. However, because existing TCP implementations [RFC5461] already interpret ICMP unreachable messages during connection establishment and abort the corresponding connection, we refrain from suggesting this.

5.6. Reaction in Steady-State

Another exploitation of ICMP unreachable messages in the context of TCP congestion control might seem appropriate, while TCP is in steady-state. As the RTT up to the router that generated the ICMP unreachable message is likely to be substantially shorter than the overall RTT to the destination, the ICMP unreachable message may very well reach the originating TCP while it is transmitting the current window of data. In case the remaining window is large, it might seem appropriate to refrain from transmitting the remaining window as there is timely evidence that it will only trigger further ICMP unreachable messages at that very router. Although this promises improvement from a wastage perspective, it may be counterproductive from a security perspective. An attacker could forge such ICMP messages, thereby forcing the originating TCP to stop sending data, very similar to the blind throughput-reduction attack mentioned in [RFC5927].

An additional consideration is the following: in the presence of multi-path routing, even the receipt of a legitimate ICMP unreachable message cannot be exploited accurately, because there is the possibility that only one of the multiple paths to the destination is suffering from a connectivity disruption, which causes ICMP unreachable messages to be sent. Then, however, there is the possibility that the path along which the connectivity disruption occurred contributed considerably to the overall bandwidth, such that a congestion response is very well reasonable. However, this is not necessarily the case. Therefore, a TCP has no means except for its inherent congestion control to decide on this matter. All in all, it seems that for a connection in steady-state, i.e., not in timeout-based loss recovery, reacting to ICMP unreachable messages in regard to congestion control is not appropriate. For the case of timeout-based retransmissions, however, there is a reasonable congestion response, which is skipping further retransmission timer backoffs because there is no congestion indication -- as described above.

6. Dissolving Ambiguity Issues Using the TCP Timestamps Option

If the TCP Timestamps option [RFC1323] is enabled for a connection, a TCP sender SHOULD use the following algorithm to dissolve the ambiguity issues mentioned in Sections 5.1, 5.2, and 5.3. In particular, both the retransmission ambiguity and the packet duplication problems are prevented by the following TCP-LCD variant. On the other hand, the false positives caused by wrapped sequence numbers cannot be completely avoided, but the likelihood is further reduced by a factor of 1/2^32, since the Timestamp Value field (TSval) of the TCP Timestamps option contains 32 bits.

Hence, implementers may choose to employ the TCP-LCD with the following modifications.

Step (1) is replaced by step (1'):

(1') Before TCP updates the variable "RTO" when it initiates timeout-based loss recovery, set the variables "BACKOFF_CNT" and "RTO_BASE", and the data structure "RETRANS_TS", as follows:

```
BACKOFF_CNT := 0;
RTO_BASE := RTO;

RETRANS_TS := [].
```

Proceed to step (R).

Step (2) is extended by step (2b):

(2b) Store the value of the Timestamp Value field (TSval) of the TCP Timestamps option included in the retransmission "RET" sent in step (R) into the "RETRANS_TS" data structure:

```
RETRANS_TS.add(RET.TSval)
```

Step (6) is replaced by step (6'):

(6') If "SEG.SEQ == SND.UNA && RETRANS_TS.exists(SEG.TSval)", i.e., if the TCP segment "SEG" eliciting the ICMP unreachable message "ICMP_DU" contains the sequence number of a retransmission, and the value in its Timestamp Value field (TSval) is valid, then

proceed to step (7');

else

proceed to step (3).

Step (7) is replaced by step (7'):

(7') Undo the last retransmission timer backoff:

```
RETRANS_TS.remove(SEQ.TSval);
BACKOFF_CNT := BACKOFF_CNT - 1;
RTO := min(RTO_BASE * 2^(BACKOFF_CNT), MAX_RTO).
```

The downside of this variant is twofold. First, the modifications come at a cost: the TCP sender is required to store the timestamps of all retransmissions sent during one timeout-based loss recovery. Second, this variant can only undo a retransmission timer backoff if the intermediate router experiencing the link outage implements [RFC1812] and chooses to include, in addition to the first 64 bits of the payload of the triggering datagram, as many bits as are needed to include the TCP Timestamps option in the ICMP unreachable message.

7. Interoperability Issues

This section discusses interoperability issues related to introducing TCP-LCD.

7.1. Detection of TCP Connection Failures

TCP-LCD may produce side effects for TCP implementations that attempt to detect TCP connection failures by counting timeout-based retransmissions. [RFC1122] states in Section 4.2.3.5 that a TCP host must handle excessive retransmissions of data segments with two thresholds, R1 and R2, that measure the number of retransmissions that have occurred for the same segment. Both thresholds might be measured either in time units or as a count of retransmissions.

Due to TCP-LCD's reversion strategy of the retransmission timer, the assumption that a certain number of retransmissions corresponds to a specific time interval no longer holds, as additional retransmissions may be performed during timeout-based-loss recovery to detect the end of the connectivity disruption. Therefore, a TCP employing TCP-LCD either MUST measure the thresholds R1 and R2 in time units or, in case R1 and R2 are counters of retransmissions, MUST convert them into time intervals that correspond to the time an unmodified TCP would need to reach the specified number of retransmissions.

7.2. Explicit Congestion Notification (ECN)

With Explicit Congestion Notification (ECN) [RFC3168], ECN-capable routers are no longer limited to dropping packets to indicate congestion. Instead, they can set the Congestion Experienced (CE) codepoint in the IP header to indicate congestion. With TCP-LCD, it may happen that during a connectivity disruption, a received ICMP unreachable message has been elicited by a timeout-based retransmission that was marked with the CE codepoint before reaching the router experiencing the link outage. In such a case, a TCP sender MUST, corresponding to Section 6.1.2 of [RFC3168], additionally reset the retransmission timer in case the algorithm undoes a retransmission timer backoff.

7.3. TCP-LCD and IP Tunnels

It is worth noting that IP tunnels, including IPsec [RFC4301], IP encapsulation within IP [RFC2003], Generic Routing Encapsulation (GRE) [RFC2784], and others, are compatible with TCP-LCD, as long as the received ICMP unreachable messages can be demultiplexed and extracted appropriately by the TCP sender during timeout-based loss recovery.

If, for example, end-to-end tunnels like IPsec in transport mode [RFC4301] are employed, a TCP sender may receive ICMP unreachable messages where additional steps, e.g., also performing decryption in step (5) of the algorithm, are needed to extract the TCP header from these ICMP messages. Provided that the received ICMP unreachable message contains enough information, i.e., SEG.SEQ is extractable, this information can still be used as a valid input for the proposed algorithm.

Likewise, if IP encapsulation like [RFC2003] is used in some part of the path between the communicating hosts, the tunnel ingress node may receive the ICMP unreachable messages from an intermediate router experiencing the link outage. Nevertheless, the tunnel ingress node may replay the ICMP unreachable messages in order to inform the TCP sender. If enough information is preserved to extract SEG.SEQ, the replayed ICMP unreachable messages can still be used in TCP-LCD.

8. Related Work

Several methods that address TCP's problems in the presence of connectivity disruptions have been proposed in literature. Some of them try to improve TCP's performance by modifying lower layers. For example, [SM03] introduces a "smart link layer", which buffers one segment for each active connection and replays these segments upon connectivity re-establishment. This approach has a serious drawback: previously stateless intermediate routers have to be modified in order to inspect TCP headers, to track the end-to-end connection, and to provide additional buffer space. This leads to an additional need for memory and processing power.

On the other hand, stateless link-layer schemes, as proposed in [RFC3819], which unconditionally buffer some small number of packets, may have another problem: if a packet is buffered longer than the maximum segment lifetime (MSL) of 2 min. [RFC0793], i.e., the disconnection lasts longer than the MSL, TCP's assumption that such segments will never be received will no longer be true, violating TCP's semantics [TCP-REXMIT-NOW].

Other approaches, like the TCP feedback-based scheme (TCP-F) [CRVP01] or the Explicit Link Failure Notification (ELFN) [HV02] inform a TCP sender about a disrupted path by special messages generated and sent from intermediate routers. In the case of a link failure, the TCP sender stops sending segments and freezes its retransmission timers. TCP-F stays in this state and remains silent until either a "route establishment notification" is received or an internal timer expires. In contrast, ELFN periodically probes the network to detect connectivity re-establishment. Both proposals rely on changes to intermediate routers, whereas the scheme proposed in this document is a sender-only modification. Moreover, ELFN does not consider congestion and may impose serious additional load on the network, depending on the probe interval.

The authors of "ad hoc TCP" (ATCP) [LS01] propose enhancements to identify different types of packet loss by introducing a layer between TCP and IP. They utilize ICMP destination unreachable messages to set TCP's receiver advertised window to zero, thus forcing the TCP sender to perform zero window probing with an exponential backoff. ICMP destination unreachable messages that arrive during this probing period are ignored. This approach is nearly orthogonal to this document, which exploits ICMP messages to undo a retransmission timer backoff when TCP is already probing. In principle, both mechanisms could be combined. However, due to security considerations, it does not seem appropriate to adopt ATCP's reaction, as discussed in Section 5.6.

Schuetz et al. [TCP-RLCI] describe a set of TCP extensions that improve TCP's behavior when transmitting over paths whose characteristics can change rapidly. Their proposed extensions modify the local behavior of TCP and introduce a new TCP option to signal locally received connectivity-change indications (CCIs) to remote peers. Upon receipt of a CCI, they re-probe the path characteristics either by performing a speculative retransmission or by sending a single segment of new data, depending on whether the connection is currently stalled in exponential backoff or transmitting in steady-state, respectively. The authors focus on specifying TCP response mechanisms; nevertheless, underlying layers would have to be modified to explicitly send CCIs to make these immediate responses possible.

9. Security Considerations

Generally, an attacker has only two attack alternatives: to generate ICMP unreachable messages to try to make a TCP modified with TCP-LCD flood the network, or to suppress legitimate ICMP unreachable messages to try to slow down the transmission rate of a TCP sender.

In order to generate ICMP unreachable messages that fit as an input for TCP-LCD, an attacker would need to guess the correct four-tuple (i.e., Source IP Address, Source TCP port, Destination IP Address, and Destination TCP port) and the exact segment sequence number of the current timeout-based retransmission. Yet, the correct sequence number is generally hard to guess, given the probability of $1/2^{32}$. Even if an attacker has information about that sequence number (i.e., the attacker can eavesdrop on the retransmissions) the impact on the network load from the attacker may be considered low, since the retransmission frequency is limited by the RTO that was computed before TCP had entered the timeout-based loss recovery. Hence, the highest probing frequency is expected to be even lower than once per minimum RTO, i.e., 1 s as specified by [RFC2988]. It is important to note that an attacker who can correctly guess the four-tuple and the segment sequence number can easily launch more serious attacks (i.e., hijack the connection), whether or not TCP-LCD is used.

There may be means by which an attacker can cause the suppression of legitimate ICMP unreachable messages (e.g., by flooding the router experiencing the link outage to trigger ICMP rate-limiting). However, even if the attacker could suppress every legitimate ICMP unreachable message, the security impact of such an attack is negligible, since the TCP sender using TCP-LCD will behave like a regular TCP would. Note that this kind of attack is indistinguishable from a router experiencing a link outage that is not sending ICMP unreachable messages at all (e.g., because of local policy).

In summary, the algorithm proposed in this document is considered to be secure.

10. Acknowledgments

We would like to thank Lars Eggert, Adrian Farrel, Mark Handley, Kai Jakobs, Ilpo Jarvinen, Enrico Marocco, Catherine Meadows, Juergen Quittek, Pasi Sarolahti, Tim Shepard, Joe Touch, and Carsten Wolff for feedback on earlier versions of this document. We also thank Michael Faber, Daniel Schaffrath, and Damian Lukowski for implementing and testing the algorithm in Linux. Special thanks go to Ilpo Jarvinen for giving valuable feedback regarding the Linux implementation.

This work has been supported by the German National Science Foundation (DFG) within the research excellence cluster Ultra High-Speed Mobile Information and Communication (UMIC), RWTH Aachen University.

11. References

11.1. Normative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

11.2. Informative References

[CRVP01] Chandran, K., Raghunathan, S., Venkatesan, S., and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks", IEEE Personal Communications vol. 8, no. 1, pp. 34-39, February 2001.

[HV02] Holland, G. and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks", Wireless Networks vol. 8, no. 2-3, pp. 275-288, March 2002.

[KP87] Karn, P. and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols", Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'87) pp. 2-7, August 1987.

[LS01] Liu, J. and S. Singh, "ATCP: TCP for mobile ad hoc networks", IEEE Journal on Selected Areas in Communications vol. 19, no. 7, pp. 1300-1315, July 2001.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3522] Ludwig, R. and M. Meyer, "The Eifel Detection Algorithm for TCP", RFC 3522, April 2003.

[RFC3782] Floyd, S., Henderson, T., and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, April 2004.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC4015] Ludwig, R. and A. Gurtov, "The Eifel Response Algorithm for TCP", RFC 4015, February 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5461] Gont, F., "TCP's Reaction to Soft Errors", RFC 5461, February 2009.

[RFC5682] Sarolahti, P., Kojo, M., Yamamoto, K., and M. Hata, "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP", RFC 5682, September 2009.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.

[SESB05] Schuetz, S., Eggert, L., Schmid, S., and M. Brunner, "Protocol enhancements for intermittently connected hosts", SIGCOMM Computer Communication Review vol. 35, no. 3, pp. 5-18, December 2005.

[SM03] Scott, J. and G. Mapp, "Link layer-based TCP optimisation for disconnecting networks", SIGCOMM Computer Communication Review vol. 33, no. 5, pp. 31-42, October 2003.

[TCP-REXMIT-NOW] Eggert, L., Schuetz, S., and S. Schmid, "TCP Extensions for Immediate Retransmissions", Work in Progress, June 2005.

[TCP-RLCI] Schuetz, S., Koutsianas, N., Eggert, L., Eddy, W., Swami, Y., and K. Le, "TCP Response to Lower-Layer Connectivity-Change Indications", Work in Progress, February 2008.

[Zh86] Zhang, L., "Why TCP Timers Don't Work Well", Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'86) pp. 397-405, August 1986.

[ZimHan09] Zimmermann, A., "Make TCP more Robust to Long Connectivity Disruptions", Proceedings of the 75th IETF Meeting slides, July 2009, <http://www.ietf.org/proceedings/75/slides/tcpm-0.pdf>.

Authors' Addresses

Alexander Zimmermann
RWTH Aachen University
Ahornstrasse 55
Aachen, 52074
Germany

Phone: +49 241 80 21422
EMail: zimmermann@cs.rwth-aachen.de

Arnd Hannemann
RWTH Aachen University
Ahornstrasse 55
Aachen, 52074
Germany

Phone: +49 241 80 21423
EMail: hannemann@nets.rwth-aachen.de