TCP Maintenance and Minor
Extensions (TCPM) WG
Internet-Draft
Intended status: Experimental
Expires: May 21, 2010

A. Zimmermann
A. Hannemann
RWTH Aachen University

November 17, 2009

Making TCP more Robust to Long Connectivity Disruptions (TCP-LCD)
draft-ietf-tcpm-tcp-lcd-00

Abstract

Disruptions in end-to-end path connectivity, which last longer than one retransmission timeout cause suboptimal TCP performance. The reason for the 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 leads in turn to a deferred detection of the re-establishment of the connection since TCP waits until the next retransmission timeout occurs before attempting the retransmission.

This document proposes a algorithm for making TCP more robust to long connectivity disruptions (TCP-LCD). The memo 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 revert strategy of the retransmission timer is specified that enables a more prompt detection of whether the connectivity to a previously disconnected peer node has been restored or not. TCP-LCD is a TCP sender-only modification that effectively improves TCP performance in presence of connectivity disruptions.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

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."

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.

This Internet-Draft will expire on May 21, 2010.

Table of Contents

1. 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 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 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" that refers to the state, which 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] extents the period 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.

2. Introduction

Connectivity disruptions can occur in many different situations. The frequency of the connectivity disruptions depends thereby on the property of the end-to-end path between the communicating hosts. While connectivity disruptions can occur in traditional wired networks too, e.g., simply due to an unplugged network cable, the likelihood of 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 exhibit 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 [I-D.schuetz-tcpm-tcp-rlci]: "short" and "long" connectivity disruptions. 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 (ACK) 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 returns. Whether or not path characteristics, like the round trip time (RTT) or the available bandwidth have changed when the connectivity returns after a disruption is another important aspect for TCP's retransmission scheme [I-D.schuetz-tcpm-tcp-rlci].

This document improves TCP's behavior in 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 additional modification to detect whenever the path characteristics remain unchanged in order to improve TCP's behavior once connectivity has been restored. Hence, TCP's basic congestion control mechanisms [RFC5681] will be unchanged.

When a long connectivity disruption occurs on a TCP connection, the TCP sender stops receiving 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 sending rate, which is based on the assumption that segment loss is indication of congestion [RFC5681]. As long as the connectivity disruption persists, TCP will repeat this procedure until the oldest outstanding segment is successfully acknowledged, or the connection times 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 60s, but may be longer: Linux for example uses 120s. If the 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 when the connectivity is re-established. Therefore, depending on when connectivity becomes available again, this can waste up to

maximum RTO of possible transmission time.

This retransmission behavior is not efficient, especially in scenarios where long connectivity disruptions are frequent. In the ideal case, a TCP would attempt a retransmission as soon as connectivity to its peer is 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 revert strategy of the retransmission timer enables, due to higher-frequency retransmissions, a prompt detection when the connectivity to a previously disconnected peer node has been restored. In the case the network allows, i.e., no congestion is present, TCP-LCD approaches the ideal behavior.

3. Connectivity Disruption Indication

As long as the queue of an intermediate router experiencing a link outage is deep enough, i.e., it can buffer all incoming packets, a connectivity disruption will only cause variation in delay, which is handled well by contemporary TCP implementations with the help of Eifel [RFC3522], [RFC4015] or Forward RTO-Recovery (F-RTO) [RFC5682]. However, if the link outage lasts too long, the router experiencing the link outage is forced to drop packets and finally to discard the according route. Means to detect such link outages comprise reacting on failed address resolution protocol (ARP) [RFC0826] queries, unsuccessful link sensing, and the like. However, this is solely in the responsibility of the respective router.

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

Provided that no other route (including no default route) to the specific destination exists, the removal of the route goes along with a notification to 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, since the reception of ICMP destination unreachable messages of these codes provide evidence that packets were dropped due to a link outage, the sending host can use them as an indication for a connectivity disruption.

Note that there are also other ICMP 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), or code 12 (host unreachable for TOS) [RFC1812]. On the other hand codes that flag hard errors are of no use for the proposed scheme, since TCP should abort the connection when those are received [RFC1122]. In the following, the term "ICMP unreachable message" is used as synonym for ICMP destination unreachable messages of code 0 or code 1.

The accurate interpretation of ICMP unreachable messages as a connectivity disruption indication is complicated by the following two peculiarities of ICMP messages. Firstly, they do not necessarily operate on the same timescale as the packets, i.e., in the given case 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 rather queues it for later delivery. Secondly, ICMP messages are subject to rate limiting, e.g., when a router drops a whole window of data due to a link outage, it will hardly send as many ICMP unreachable messages as it dropped TCP segments. Depending on the load of the router it may even send no ICMP unreachable messages at all. Both peculiarities originate from [RFC1812].

Fortunately, according to [RFC0792] ICMP unreachable messages are obliged to contain in their body the entire Internet Protocol (IP) header [RFC0791] of the datagram eliciting the ICMP unreachable messages plus the first 64 bits of the payload of that datagram. This allows the sending host to match the ICMP error message to the transport that elicited it. RFC 1812 [RFC1812] augments the requirements and states that ICMP messages should contain as much of the original datagram as possible without the length of the ICMP datagram exceeding 576 bytes. Therefore, in case of TCP, at least the source port number, the destination port number, and the 32-bit TCP sequence number are included. Thus, this allows the originating TCP to demultiplex the received ICMP message and to identify the connection which an ICMP unreachable message is reporting an error about. Moreover, it can identify which segment of the respective connection triggered the ICMP unreachable message, provided that there are not several segments in-flight with the same sequence number (see Section 5.1).

A connectivity disruption indication in 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 instead was successfully delivered to the temporary end-point of the employed path, i.e., the reporting router. It therefore did not

witness any congestion at least on that very part of the path that was traveled by both, the TCP segment eliciting the ICMP unreachable message as well as the ICMP unreachable message itself.

4. Connectivity Disruption Reaction

Section 4.1 gives 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 the prompt detection when the 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 to increase the TCP's retransmission frequency by undoing one retransmission timer backoff whenever an ICMP unreachable message reports on the 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 using 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.

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

In rule (2.5) RFC 2988 [RFC2988] provides the option to place a maximum value on the RTO. When a TCP implements this rule to provide an upper bound for the RTO, the rule SHOULD 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

SHOULD be initialized with this upper limit. Otherwise, if the RTO is unbounded, the "MAX_RTO" variable SHOULD 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 in case 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 the behavior that a standard TCP must execute at this point in case the retransmission timer is expired. In particular if RFC 2988 [RFC2988] is used, steps (5.4) - (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

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

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

or 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" carries 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 the standard TCP behavior that must be executed at this point in the 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 the retransmission timer is actually backed off or not. 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 of no use and therefore 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) allows, a TCP SHOULD undo one backoff for each ICMP unreachable message reporting an error on a retransmission. To decide if an ICMP unreachable message reports on a retransmission, the sequence number therein is exploited (step 5, step 6). The undo is performed by re-calculating 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 expired already (step 8). Then, a TCP SHOULD retransmit immediately, i.e., an ICMP message clocked retransmission. In case the shortened retransmission timer has not expired yet, TCP MUST wait accordingly.

5. Discussion of TCP-LCD

TCP-LCD takes caution to only react to connectivity disruption indications in 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 yet expire since the last received acceptable ACK. Thereby, the algorithm triggers by definition only in the case of long connectivity disruptions.

Only such ICMP unreachable messages that report on the sequence number of a retransmission, i.e., report on 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 instead were successfully delivered to the temporary end-point of the employed path, i.e., the reporting router. In other words, there is no witness for any congestion at least on that very part of the path that was traveled by both, the TCP segment eliciting the ICMP unreachable message as well as the ICMP unreachable message itself.

However, there are some situations where TCP-LCD makes a false decision and undoes a retransmission timer backoff wrongly. This can happen, albeit the received ICMP unreachable message reports on 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 5.5 describes the motivation for not reacting on 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 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 revert strategy of the given algorithm suffers from a form of retransmission ambiguity, too. In contrast to the aforementioned 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 which TCP segment, i.e., the original transmission or any of the retransmissions an ICMP unreachable message reports on.

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

5.2. Wrapped Sequence Numbers

Besides the ambiguity if a received ICMP unreachable message refers to the original transmission or to any of the retransmissions, there is another source of ambiguity about the TCP sequence numbers contained in ICMP unreachable messages. For high bandwidth paths like modern gigabit links the sequence space may wrap rather quickly, thereby allowing the possibility that delayed ICMP unreachable messages - a router dropping packets due to a link outage is not obliged to send ICMP unreachable messages in a timely manner [RFC1812] - may coincidentally fit as valid input in the proposed scheme. As a result, the scheme may undo retransmission timer backoffs wrongly. Chances for this to happen are minuscule, since a particular ICMP 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 R 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 any reason, router R delays all corresponding ICMP unreachable messages, so that the TCP sender backoffs the retransmission timer normally without any undoing. At the end of the connectivity disruption, the TCP sender eventually detects the re-establishment, 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 router R emits the delayed ICMP unreachable messages now, the TCP sender would undo retransmission timer backoffs wrongly. 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 by an intermediate router R due to a link outage. For each segment in-flight, router R may generates an ICMP unreachable message, however due to some reason it delays them. Once the link outage is over and the connection is 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 router R emits the delayed ICMP unreachable messages now, one spurious undoing of the retransmission timer backoff is possible, if firstly the TCP segment number contained in ICMP unreachable messages matches the current SND.UNA, and secondly 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 the case an intermediate router duplicates packets, a TCP sender may receive more ICMP unreachable messages during timeout-based loss recovery than it actually has 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 messages could hide the congestion loss of some retransmissions or ICMP messages and the algorithm may undo retransmission timer backoffs wrongly. 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 could argue that if an ICMP unreachable message arrives for a timeout-based retransmission, the RTO should 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 allow for a much higher probing frequency based on the round trip time up to the router where the connectivity is disrupted. However, we believe the current scheme provides a good trade-off between conservative behavior and fast detection of connectivity re-establishment.

5.5. Reaction in Steady-State

Another exploitation of ICMP unreachable messages in the context of TCP congestion control might seem appropriate in case the ICMP unreachable message is received while TCP is in steady-state and the message refers to a segment from within the current window of data. As the RTT up to the router, which generates 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 the 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 [I-D.ietf-tcpm-icmp-attacks].

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 option 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 on 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 (the Safe Variant)

Given that the TCP Timestamps option [I-D.ietf-tcpm-1323bis] is enabled for a connection, a TCP sender MAY 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 can not 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 implement 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(SEQ.TSval)", i.e., if the TCP segment "SEG" eliciting the ICMP unreachable message "ICMP_DU" carries 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 the safe variant is twofold. Firstly, 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. Secondly, the safe variant can only undo a retransmission timer backoff, if the intermediate router experiencing the link outage implements [RFC1812] and chooses to include as many more than the first 64 bits of the payload of the triggering datagram, 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 have side-effects on TCP implementations, which attempt to detect TCP connection failures by counting timeout-based retransmissions. RFC 1122 [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 measuring the amount of retransmission that has occurred for the same segment. Both thresholds might either be measured in time units or as a count of retransmissions.

Due to TCP-LCD's revert strategy of the retransmission timer, the assumption that a certain number of retransmissions corresponds to a specific time interval no longer holds true, 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 SHOULD measure the thresholds R1 and R2 in time units or in case R1 and R2 are counters of retransmissions SHOULD convert them into time intervals, which correspond to the time an unmodified TCP would need to reach the specified number of retransmissions.

7.2. Explicit Congestion Notification

By the use of Explicit Congestion Notification (ECN) [RFC3168] ECN-capable routers are no longer limited to dropping packets as congestion indication. Instead they can set the Congestion Experienced (CE) codepoint in the IP header of packets to indicate congestion. Concerning TCP-LCD there is the option 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, we suggest in the case the algorithm undoes a retransmission timer backoff, the TCP sender SHOULD additionally reset the retransmission timer.

7.3. ICMP for IP version 6

RFC 4443 [RFC4443] specifies the Internet Control Message Protocol (ICMPv6) to be used with the Internet Protocol version 6 (IPv6)

[RFC2460]. From TCP-LCD's point of view, it is important to notice that for IPv6, the payload of an ICMPv6 error messages has to include as many bytes from the IPv6 datagram that elicited the ICMPv6 error message as possible without making the error message exceed the minimum IPv6 MTU (1280 bytes) [RFC4443]. Thus, more information is available for TCP-LCD as in the case of IPv4.

The counterpart of the ICMPv4 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]. Like the IPv4 case, 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. As a result, TCP-LCD can employ this ICMPv6 error messages as connectivity disruption indication, too.

7.4. TCP-LCD and IP Tunnels

It is worth noting that IP tunnels, including IPsec [RFC4301], IP in 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., decrypting in step (5) of the algorithm is needed to extract the TCP header from these ICMP messages. Provided that the received ICMP unreachable message contains enough information, i.e., SEQ.SEG is extractable, these information MAY 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, instead of the TCP sender, 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 SEQ.SEG, the replayed ICMP unreachable messages MAY still be used in TCP-LCD.

8. Related Work

In literature there are several methods that address TCP's problems in the presence of connectivity disruptions. 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 replaying these segments on 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. These lead all in all to an additional need of memory and processing power.

On the other hand stateless link layer schemes, like 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 MSL, TCP's assumption that such segments will never be received will no longer be true, violating TCP's semantics [I-D.eggert-tcpm-tcp-retransmit-now].

Other approaches like 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 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 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 and thus forcing the TCP sender to perform zero window probing with a exponential backoff. ICMP destination unreachable messages, which 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 5.5.

Schuetz et al. describe in [I-D.schuetz-tcpm-tcp-rlci] a set of TCP extensions that improve TCP's behavior when transmitting over paths whose characteristics can change on short time-scales. 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 reception 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. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

The algorithm proposed in this document is considered to be secure. For example an attacker, who already guessed the correct four-tuple (i.e., Source IP Address, Source TCP port, Destination IP Address, and Destination TCP port), can still not make a TCP modified with TCP-LCD to flood the network just by sending forged ICMP unreachable messages in an attempt to maliciously shorten the retransmission timer. The attacker additionally would need to guess the correct segment sequence number of the current timeout-based retransmission, with a probability of at most 1/2^32. Even in the case of man-in-the-middle attacks, i.e., attacks performed in scenarios in which the attacker can sniff the retransmissions, the impact on network load is considered to be low, since the retransmission frequency is limited by the RTO that was computed before TCP has entered the timeout-based loss recovery. Hence, the highest probing frequency is expected to be even lower than once per minimum RTO, i.e. 1s as specified by [RFC2988].

11. Acknowledgments

We would like to thank Ilpo Jarvinen, Pasi Sarolahti, Timothy 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.

12. References

12.1. Normative References

[I-D.ietf-tcpm-1323bis]
Borman, D., Braden, R., and V. Jacobson, "TCP Extensions for High Performance", draft-ietf-tcpm-1323bis-01 (work in progress), March 2009.

[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.

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

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

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

12.2. 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.

[I-D.eggert-tcpm-tcp-retransmit-now]
Eggert, L., "TCP Extensions for Immediate Retransmissions", draft-eggert-tcpm-tcp-retransmit-now-02 (work in progress), June 2005.

[I-D.ietf-tcpm-icmp-attacks]
Gont, F., "ICMP attacks against TCP", draft-ietf-tcpm-icmp-attacks-06 (work in progress), August 2009.

[I-D.schuetz-tcpm-tcp-rlci]
Schuetz, S., Koutsianas, N., Eggert, L., Eddy, W., Swami, Y., and K. Le, "TCP Response to Lower-Layer Connectivity-

Change Indications", draft-schuetz-tcpm-tcp-rlci-03 (work in progress), February 2008.

[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, 2001 July.

[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.

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

[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.

[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.

[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.

[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.

[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.

Appendix A. Changes from previous versions of the draft

A.1. Changes from draft-zimmermann-tcp-lcd-02

- Incorporated feedback submitted by Ilpo Jarvinen. <http://www.ietf.org/mail-archive/web/tcpm/current/msg04841.html>
- Incorporated feedback submitted by Pasi Sarolahti. <http://www.ietf.org/mail-archive/web/tcpm/current/msg04870.html>
- Incorporated feedback submitted by Joe Touch. <http://www.ietf.org/mail-archive/web/tcpm/current/msg04895.html>

<http://www.ietf.org/mail-archive/web/tcpm/current/msg04900.html>

- Extended and reorganized the discussion (Section 5):
  - Every discussion item got its own title, so that we have a better overview.
  - Extended Retransmission Ambiguity section. Added also some references to the historical retransmission ambiguity problem.
  - Heavily extended discussion about wrapped sequence numbers (see Joe's comments).
  - Described the influence of packet duplication on the algorithm (Thanks to Ilpo).
  - The section "Protecting Against Misbehaving Routers" is not a subsection anymore. Moreover, the section was renamed to "Dissolving Ambiguity Issues" and has now real content.
- An interoperability issues section (Section 7) was added. In particular comments to ECN, ICMPv6, and to the two thresholds R1 and R2 of [RFC1122] (Section 4.2.3.5) were added.
- Miscellaneous editorial changes. In particular, the algorithm has a name now: TCP-LCD.

A.2. Changes from draft-zimmermann-tcp-lcd-01

- The algorithm in Section 4.2 was slightly changed. Instead of reverting the last retransmission timer backoff by halving the RTO, the RTO is recalculated with help of the "BACKOFF_CNT" variable. This fixes an issue that occurred when the retransmission timer was backed off but bounded by a maximum value. The algorithm in the previous version of the draft, would have "reverted" to half of that maximum value, instead of using the value, before the RTO was doubled (and then bounded).
- Miscellaneous editorial changes.

A.3. Changes from draft-zimmermann-tcp-lcd-00

- Miscellaneous editorial changes in Section 1, 2 and 3.
- The document was restructured in Section 1, 2 and 3 for easier reading. The motivation for the algorithm is changed according TCP's problem to disambiguate congestion from non-congestion loss.

- Added Section 4.1.
- The algorithm in Section 4.2 was restructured and simplified:
  - The special case of the first received ICMP destination unreachable message after an RTO was removed.
  - The "BACKOFF_CNT" variable was introduced so it is no longer possible to perform more reverts than backoffs.
- The discussion in Section 5 was improved and expanded according to the algorithm changes.

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