Internet Engineering Task Force A. Zimmermann
Internet-Draft A. Hannemann
Intended status: Experimental RWTH Aachen University
Expires: February 27, 2010 August 26, 2009

Make TCP more Robust to Long Connectivity Disruptions
draft-zimmermann-tcp-lcd-02

Status of this Memo

This Internet-Draft will expire on February 27, 2010.

Copyright Notice

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 connectivity disruptions as a sign of congestion, resulting in repeated backoffs of the retransmission timer. 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 describes how standard ICMP messages can be exploited to disambiguate true congestion loss from non-congestion loss caused by long 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. The specified algorithm is a TCP sender-only modification that effectively improves TCP performance in presence of connectivity disruptions.

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].

As defined in [RFC0793], the term "acceptable acknowledgment (ACK)" refers to a TCP segment that acknowledges previously unacknowledged data. The Transmission Control Protocol (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.

We use both the term "retransmission timer" and the term "retransmission timeout (RTO)" as defined in [RFC2988].

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.

According to Schuetz et. al. [I-D.schuetz-tcpm-tcp-rlci] connectivity disruptions can be classified into two groups: "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 will focus on TCP's behavior in face of long connectivity disruptions in the time "before" connectivity is restored. In particular this memo does not describe any additional modification to detect if the path characteristics remain unchanged in order to improve TCP's behavior "after" connectivity is restored. Therefore, TCP's congestion control mechanisms [I-D.ietf-tcpm-rfc2581bis] 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 loss is indication of congestion [I-D.ietf-tcpm-rfc2581bis]. As long as the connectivity disruption persists, TCP will repeat the 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 or networks like wireless (multi-hop) networks where connectivity disruptions are frequent. In the ideal case, TCP would attempt a retransmission as soon as connectivity to its peer is re-established. This document describes how the standard Internet Control Message Protocol (ICMP) can be exploited to identify non-congestion loss caused by connectivity disruptions. An revert strategy of the retransmission timer is specified that enables, due to higher-frequency retransmissions, a prompt detection of whether connectivity to a previously disconnected peer node has been restored. The specified scheme is a TCP sender-only modification, i.e., neither intermediate routers nor the TCP receiver have to be modified. Furthermore, in the case the network allows, i.e., no congestion is present, the proposed algorithm 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] or forward RTO (F-RTO) [I-D.ietf-tcpm-rfc4138bis]. 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.

The removal of the route usually goes along with a notification to the corresponding TCP sender about the dropped packets via ICMP destination unreachable messages of code 0 (net unreachable) or code 1 (host unreachable) [RFC1812]. Therefore, since ICMP destination unreachable messages of these codes provide evidence that packets were dropped due to a link outage, they can be used by a TCP 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 side 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 an 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, which 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 Internet Protocol (IP) header [RFC0791] of the datagram eliciting the ICMP unreachable messages plus the first 64 bits of the payload of that datagram. Hence, in case of TCP both port numbers and the sequence number are included. This allows the originating TCP to identify the connection which an ICMP unreachable message is reporting an error about. Moreover, it allows the originating TCP to 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. This may very well be the case when TCP is recovering lost segments (see Section 4.3).

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 successful 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 which was traveled by both, the TCP segment eliciting the ICMP unreachable message as well as the ICMP unreachable message itself.

4. Connectivity Disruption Reaction

In Section 4.1 the basic idea of the algorithm is given. The complete algorithm is specified in Section 4.2. In Section 4.3 the algorithm is discussed in detail.

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. The proposed algorithm exploits standard ICMP unreachable messages to increase the TCP's retransmission frequency during timeout-based loss recovery by undoing one retransmission timer backoff whenever an ICMP unreachable message reports on a presumably lost retransmission.

This approach has the advantage of appropriately reducing the probing rate in case of congestion. If either the (re-)transmission itself, or the corresponding ICMP message is dropped the conventional backoff

is performed and not undone, effectively halving the probing rate.

4.2. The Algorithm

A TCP sender using RFC 2988 [RFC2988] to compute TCP's retransmission timer MAY employ the following scheme to avoid over-conservative backoffs of the retransmission timer in case of long connectivity disruptions. If a TCP sender does implement the scheme, the following steps MUST be taken, but only upon initiation of a timeout-based loss recovery, i.e., upon the first timeout of the oldest outstanding segment (SND.UNA). The algorithm MUST NOT be re-initiated after a timeout-based loss recovery has already been started but not completed. In particular, it 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 the scheme. We envision that the scheme could be easily adapted to other algorithms than RFC 2988. However, we leave this as future work.

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 the variable RTO gets updated when timeout-based loss recovery is initiated, set the variable "Backoff_cnt" and the variable "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) If the retransmission timer was backed off in the previous step (R), then increment the variable "Backoff_cnt" by one to account for the new backoff

```
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., an undoing of the last retransmission timer backoff is allowed, then

proceed to step (5);

else

proceed to step (3).

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

SEG := Extract(ICMP_DU).

(6) If "SEG.SEQ == SND.UNA", i.e., the ICMP unreachable ICMP_DU message reports on the oldest outstanding segment, then undo the last retransmission timer backoff

Backoff_cnt := Backoff_cnt - 1;
RTO := RTO_base * 2^(Backoff_cnt).

(7) If the retransmission timer expires due to the undoing in the previous step (6), 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, back offs the retransmission timer and retransmits the first unacknowledged segment (step R) [I-D.ietf-tcpm-rfc2581bis] [RFC2988].

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). If a maximum value is placed on the RTO (rule 2.5 in [RFC2988]) and that maximum value is already reached the TCP will not backoff the retransmission timer in this step and thus "Backoff_cnt" MUST NOT be incremented. However, the "last step" to reach this maximum RTO is still considered as a backoff in the scope of this algorithm and "Backoff_cnt" MUST be incremented, even if the RTO is not strictly doubled.

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 side if the first received packet after the retransmission(s) is an ICMP unreachable message (step 3c), a TCP SHOULD if allowed (step 4) 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 done by re-calculating the RTO with the previously reduced "Backoff_cnt". This calculation explicitly matches the exponential backoff specified in [RFC2988] (rule 5.5).

Upon receipt of an ICMP unreachable message which legitimately undoes one backoff there is the possibility that this new started retransmission timer has expired already (step 7). Then, a TCP SHOULD retransmit immediately, i.e., an ICMP message clocked retransmission. In case the new started retransmission timer has not expired yet, TCP MUST wait accordingly.

4.3. Discussion

It is important to note that the proposed algorithm only reacts to connectivity disruption indications in form of ICMP destination unreachable messages during the phase of RTO induced loss recovery. That is, TCP’s behavior is not altered when no ICMP unreachable messages are received, or the retransmission timer of the TCP sender did not yet expire since the last successfully received ACK. Thereby

the algorithm is by definition only triggered in the case of long connectivity disruptions.

Only such ICMP unreachable messages which are reporting on the sequence number of the retransmission (SND.UNA) are evaluated by the proposed algorithm. All other ICMP unreachable messages are ignored. If an ICMP unreachable message arrives for a retransmission it provides evidence that neither the retransmission nor the corresponding ICMP unreachable message itself did experience any congestion. In other words, it has been proved that the retransmission was not lost due to congestion, but due to a connectivity disruption instead.

One could argue, that if an ICMP unreachable message arrives for an RTO induced retransmission, the RTO should be reset, and the next retransmission sent out immediately similar to what is done when an ACK arrives after an RTO induced recovery phase. This would allow for a much higher probing frequency based on the round trip time of the router where the connectivity is disrupted. However, we consider our proposed scheme a good trade off between conservative behavior and a fast detection of connectivity re-establishment.

Of course there is an ambiguity on which (re-)transmission an ICMP unreachable message reports. However, for our purposes it is not considered to be problem, because the assumption that such an ICMP message provides evidence that one link loss was wrongly considered as a congestion loss, still holds. There is also the option to make use of the timestamps option to obtain a more strict mapping between segments and ICMP messages (see Section 4.3).

Besides the ambiguity if the first unacknowledged sequence number refers to the original transmission or to any of the retransmissions, there is another source of ambiguity about the sequence numbers contained in the ICMP unreachable messages. For high bandwidth paths like modern gigabit links the sequence space may wrap rather quickly, thereby allowing the possibility that a late ICMP unreachable message reporting on an old error may coincidentally fit as input in the scheme explained above. As a result, the scheme would wrongly undo one backoff. Chances for this to happen are minuscule, since a particular ICMP message would need to contain the exact sequence number of SND.UNA, while at the same TCP is coincidentally in timeout-based loss recovery. Moreover, as the scheme is tailored most conservatively no threat to the network from this issues may arise.

Finally, the scheme explicitly does not call for a differentiation of ICMP unreachable messages originating from different routers, as the evidence of no congestion still holds even if the reporting router

changed.

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 might seem appropriate from a wastage perspective, it may be counterproductive from a security perspective since ICMP message are easy to spoof, thereby allowing an easy attack to the TCP by simply forging such ICMP messages.

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 RTO induced recovery, reacting on ICMP unreachable messages in regard to congestion control is not appropriate. For the case of RTO-based retransmissions, however, there is a reasonable congestion response, which is skipping further backoffs of the retransmission timer because there is no congestion indication - as described above.

4.4. Protecting Against Misbehaving Routers (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 protect against misbehaving routers.

5. 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" that buffers one segment for each ongoing 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 the TCP sender about a disrupted path by special messages generated from intermediate routers. In case of a link failure they stop sending segments and freeze TCP's 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 also 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 4.3.

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.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

The proposed algorithm is considered to be secure. For example an attacker cannot make a TCP modified with proposed scheme flood the network just by sending forged ICMP unreachable messages to attempt to maliciously shorten the retransmission timer. An attacker would need to guess the correct sequence number of the current retransmission, which seems very unlikely. Even in case of an omniscient attacker, the impact on network load would be low, since the retransmission frequency is limited by the RTO which was computed before TCP has entered the timeout-based loss recovery. (The highest probing frequency is expected to be even lower than once per minimum RTO, that is 1s as specified by [RFC2988].)

8. Acknowledgments

We would like to thank Timothy Shepard and Joe Touch for feedback on earlier versions of this draft. We also thank Michael Faber, Daniel Schaffrath, and Damian Lukowski for implementing and testing the algorithm in Linux. Special thanks go to Ilpo Jarvinen, who gave valuable feedback regarding the Linux implementation.

This document was written with the xml2rfc tool described in [RFC2629].

9. References

9.1. 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.

[I-D.ietf-tcpm-rfc2581bis]
Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", draft-ietf-tcpm-rfc2581bis-07 (work in progress), July 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.

[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.

9.2. 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-rfc4138bis]
Sarolahti, P., Kojo, M., Yamamoto, K., and M. Hata, "Forward RTO-Recovery (F-RTO): An Algorithm for Detecting Spurious Retransmission Timeouts with TCP", draft-ietf-tcpm-rfc4138bis-04 (work in progress), October 2008.

[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.

[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.

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

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

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

[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.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, April 2007.

[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.

Appendix A. TODO list

- Extend the Security Sections 4.4 and 7.
- Extend discussion in Section 4.3
  - ICMPv6. See [RFC4443] and [RFC4884].
  - Explicit Congestion Notification (ECN).
  - More about congestion in general.
- Mention the possible side-effect on TCP implementations that measure the thresholds R1 and R2 (Section 4.2.3.5 of [RFC1122]) as a count of retransmissions instead of time units.
- Discuss the influence of packet duplication on the algorithm (Thanks to Ilpo).

Appendix B. Changes from previous versions of the draft

B.1. Changes from draft-zimmermann-tcp-lcd-01

- The algorithm in Section 4.2 was slightly changed. Instead of reverting the RTO by halving it, it 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.
- Extended the TODO list (Appendix A).

B.2. 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 4.3 was improved and expanded according to the algorithm changes.
- Added Section 4.4.

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