

TCP Maintenance and Minor
Extensions (tcpm)
Internet-Draft
Expires: August 20, 2006

F. Gont
UTN/FRH
February 16, 2006

TCP's Reaction to Soft Errors
draft-ietf-tcpm-tcp-soft-errors-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of 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/lid-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 August 20, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document discusses the problem of long delays between connection establishment attempts that may arise in a number of scenarios, including that in which dual stack nodes that have IPv6 enabled by default are deployed in IPv4 or mixed IPv4 and IPv6 environments. Additionally, it describes a modification to TCP's reaction to soft errors that has been implemented in a variety of TCP/IP stacks to help overcome this problem.

Table of Contents

1. Introduction	3
2. Error Handling in TCP	3
2.1. Reaction to Hard Errors	4
2.2. Reaction to Soft Errors	4
3. Problems that may arise from TCP's reaction to soft errors	5
3.1. General Discussion	5
3.2. Problems that may arise with Dual Stack IPv6 on by Default	5
4. A workaround for long delays between connection-establishment attempts	6
5. Possible drawbacks	7
5.1. Non-deterministic transient network failures	7
5.2. Deterministic transient network failures	7
6. Future work	7
7. Security Considerations	8
8. Acknowledgements	8
9. Contributors	8
10. References	9
10.1. Normative References	9
10.2. Informative References	9
Appendix A. Other possible solutions	10
A.1. A more conservative approach	10
A.2. Asynchronous Application Notification	11
A.3. Issuing several connection requests in parallel	11
Appendix B. Change log (to be removed before publication of the document as an RFC)	12
B.1. Changes from draft-gont-tcpm-tcp-soft-errors-02	12
B.2. Changes from draft-gont-tcpm-tcp-soft-errors-01	12
B.3. Changes from draft-gont-tcpm-tcp-soft-errors-00	12
Author's Address	13
Intellectual Property and Copyright Statements	14

1. Introduction

The handling of network failures can be separated into two different actions: fault isolation and fault recovery. Fault isolation is the actions that hosts and routers take to determine that there is some network failure. Fault recovery, on the other hand, is the actions that hosts and routers will perform to isolate and survive a network failure. [RFC0816]

In the Internet architecture, the Internet Control Message Protocol (ICMP) [RFC0792] is used to perform the fault isolation function, that is, to report network error conditions to the hosts sending datagrams over the network.

When a host is signalled of a network error, there is still the issue of what to do to let communication survive, if possible, the network failure. The fault recovery strategy may depend on the type of network failure taking place, and the time the error condition is detected.

This document analyzes the fault recovery strategy of TCP [RFC0793], and the problems that may arise due to TCP's policy of reaction to soft errors. Among others, it analyzes the problems that may arise in scenarios where dual stack nodes that have IPv6 enabled by default are deployed in IPv4 or mixed IPv4 and IPv6 environments.

Additionally, it documents a modification to TCP's policy of reaction to ICMP messages indicating "soft errors", that has been implemented in a variety of TCP/IP stacks to help overcome the problems discussed in this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Error Handling in TCP

Network errors can be divided into soft and hard errors. Soft errors are considered to be transient network failures, which will hopefully be solved in the near term. Hard errors, on the other hand, are considered to reflect permanent network error conditions, which are unlikely to be solved in the near future.

Therefore, it may make sense for the fault recovery action to be different depending on the type of error being detected.

When there is a network failure that's not signalled to the sending

host, such as a gateway corrupting packets, TCP's fault recovery action is to repeatedly retransmit the segment until either it gets acknowledged, or the connection times out. In case the connection times out before the segment is acknowledged, TCP won't be able to provide more information than the timeout condition.

In case a host does receive an ICMP error message referring to an ongoing TCP connection, the IP layer will pass this message up to corresponding TCP instance to raise awareness of the network failure. [RFC1122]

TCP's reaction to ICMP messages will depend on the type of error being signalled.

2.1. Reaction to Hard Errors

When receiving a segment with the RST bit set, or an ICMP error message indicating a hard error condition, TCP will simply abort the corresponding connection, regardless of the state the connection is in.

The "Requirements for Internet Hosts -- Communication Layers" RFC [RFC1122] states, in section 4.2.3.9, that TCP SHOULD abort connections when receiving ICMP error messages that indicate hard errors. This policy is based on the premise that, as hard errors indicate network error conditions that won't change in the near term, it will not be possible for TCP to recover from this type of network failure.

2.2. Reaction to Soft Errors

If an ICMP error message is received that indicates a soft error, TCP will just record this information [Stevens], and repeatedly retransmit the data until either they get acknowledged or the connection times out.

The "Requirements for Internet Hosts -- Communication Layers" RFC [RFC1122] states, in section 4.2.3.9, that TCP MUST NOT abort connections when receiving ICMP error messages that indicate soft errors. This policy is based on the premise that, as soft errors are transient network failures that will hopefully be solved in the near term, one of the retransmissions will succeed.

In case the connection timer expires, and an ICMP error message has been received before the timeout, TCP will use this information to provide the user with a more specific error message. [Stevens]

This handling of soft errors exploits the valuable feature of the

Internet that for many network failures, the network can be dynamically reconstructed without any disruption of the endpoints.

3. Problems that may arise from TCP's reaction to soft errors

3.1. General Discussion

Even though TCP's fault recovery strategy in the presence of soft errors allows for TCP connections to survive transient network failures, there are scenarios in which this policy may cause undesirable effects.

For example, consider the case in which an application on a local host is trying to communicate with a destination whose name resolves to several IP addresses. The application on the local host will try to establish a connection with the destination host, cycling through the list of IP addresses, until one succeeds [RFC1123]. Suppose that some (but not all) of the addresses in the returned list are permanently unreachable. If they are the first IP addresses in the list, the application will usually try to use these addresses first.

As discussed in Section 2, this unreachability condition may or may not be signalled to the sending host. If the local TCP is not signalled of the error condition, there is very little that can be done other than repeatedly retransmit the SYN segment, and wait for the existing timeout mechanism in TCP, or an application timeout, to be triggered. However, even if unreachability is signalled by some intermediate router to the local TCP by means of an ICMP error message, the local TCP will just record the error message and will still repeatedly retransmit the SYN segment until the connection timer expires. The "Requirements For Internet Hosts -- Communication Layers" RFC [RFC1122] states that this timer MUST be large enough to provide retransmission of the SYN segment for at least 3 minutes. This would mean that the application on the local host would spend several minutes for each unreachable address it uses for trying to establish a TCP connection. These long delays between connection establishment attempts would be inappropriate for interactive applications such as the web. [Shneiderman] [Thadani]

3.2. Problems that may arise with Dual Stack IPv6 on by Default

Another scenario in which this type of problem may occur is that where dual stack nodes that have IPv6 enabled by default are deployed in IPv4 or mixed IPv4 and IPv6 environments, and the IPv6 connectivity is non-existent [I-D.ietf-v6ops-v6onbydefault].

As discussed in [I-D.ietf-v6ops-v6onbydefault], there are two

possible variants of this scenario, which differ in whether the lack of connectivity is signalled to the sending node, or not.

In cases where packets sent to a destination are silently dropped and no ICMPv6 [RFC2463] errors are generated, there is very little that can be done other than waiting for the existing connection timeout mechanism in TCP, or an application timeout, to be triggered.

In cases where a node has no default routers and Neighbor Unreachability Detection (NUD) fails for destinations assumed to be on-link, or where firewalls or other systems that enforce scope boundaries send ICMPv6 errors, the sending node will be signalled of the unreachability problem. However, as discussed in Section 2.2, TCP implementations will not abort connections when receiving ICMP error messages that indicate soft errors.

4. A workaround for long delays between connection-establishment attempts

As discussed in Section 1, it may make sense for the fault recovery action to depend not only on the type of error being reported, but also on the time the error is reported. For example, one could infer that when an error arrives in response to opening a new connection, it is probably caused by opening the connection improperly, rather than by a transient network failure. [RFC0816]

A variety of TCP/IP stacks have modified TCP's reaction to soft errors, to make it abort a connection in the SYN-SENT or the SYN-RECEIVED state if it receives an ICMP "Destination Unreachable" message that indicates a soft error about that connection.

The "Requirements for Internet Hosts -- Communication Layers" RFC [RFC1122] states, in section 4.2.3.9., that the ICMP "Destination Unreachable" messages that indicate soft errors are ICMP codes 0 (network unreachable), 1 (host unreachable), and 5 (source route failed). Even though ICMPv6 didn't exist when [RFC1122] was written, one could extrapolate the concept of soft errors to ICMPv6 Type 1 Codes 0 (no route to destination) and 3 (address unreachable).

It must be noted that this behaviour violates section 4.2.3.9 of [RFC1122], since it states that as these Unreachable messages indicate soft error conditions, TCP MUST NOT abort the corresponding connection.

This workaround has been implemented, for example, in the Linux kernel since version 2.0.0 (released in 1996) [Linux]. Appendix A.1 discusses a more conservative approach than the one introduced in

this section.

5. Possible drawbacks

The following subsections discuss some of the possible drawbacks arising from the use of the modification to TCP's reaction to soft errors described in Section 4.

5.1. Non-deterministic transient network failures

In case there's a transient network failure affecting all of the addresses returned by the name-to-address translation function, all destinations could be unreachable for some short period of time. In such a scenario, the application could quickly cycle through all the IP addresses in the list and return an error, when it could have let TCP retry a destination a few seconds later, when the transient problem could have disappeared.

However, it must be noted that non-interactive applications, such as a Mail Transfer Agent (MTA), usually must implement application-layer retry mechanisms, and thus are able to handle these scenarios appropriately. For interactive applications, the user would likely not be satisfied with a connection attempt that succeeds only after several seconds, anyway. [Guynes]

5.2. Deterministic transient network failures

There are some scenarios in which transient network failures could be deterministic. For example, consider the case in which upstream network connectivity is triggered by network use. In this scenario, the connection triggering the upstream connectivity would deterministically receive ICMP Destination Unreachables while the upstream connectivity is being activated, and thus would be aborted.

As discussed in Section 5.1, applications usually implement mechanisms to handle these scenarios appropriately. Also, connection attempts are usually preceded by a UDP-based DNS name-to-address lookup. Thus, unless the name-to-address mapping has been cached by a local nameserver or resolver, it will be the DNS query that will trigger the upstream network connectivity, and thus the corresponding connection will not be aborted.

6. Future work

A Higher-Level API would be useful for isolating applications from protocol details. The API could contain the intelligence required to

resolve the hostname, try each destination address, etc. One could even argue that this document wouldn't have existed if application programmers had been using a Higher-Level API. However, such an API would need to be designed, standardized, implemented, deployed, and documented even before application programmers start (if ever) to use it.

7. Security Considerations

This document describes a modification to TCP's reaction to soft errors that has been implemented in a variety of TCP/IP stacks. This modification makes TCP abort a connection in the SYN-SENT or the SYN-RECEIVED states when it receives an ICMP "Destination Unreachable" message that indicates a "soft error" about that connection. While this modification could be exploited to reset valid connections, it must be noted that this behaviour is meant only for connections in the SYN-SENT or the SYN-RECEIVED states, and thus the window of exposure is very short.

In any case, it must be noted that the workaround discussed in this document neither strengthens nor weakens TCP's resistance to attack. An attacker wishing to reset ongoing TCP connections could perform the attack by sending any of the ICMP error messages that indicate "hard errors", not only for connections in the SYN-SENT or the SYN-RECEIVED states, but for connections in any state.

A discussion of the use of ICMP to perform a variety of attacks against TCP, and a number of counter-measures that eliminate or greatly minimize the impact of these attacks can be found in [I-D.gont-tcpm-icmp-attacks].

A discussion of the security issues arising from the use of ICMPv6 can be found in [RFC2463].

8. Acknowledgements

The author wishes to thank Michael Kerrisk, Eddie Kohler, Mika Liljeberg, Pasi Sarolahti, Pekka Savola, and Joe Touch, for contributing many valuable comments.

9. Contributors

Mika Liljeberg was the first to describe how their implementation treated soft errors. Based on that, the solution discussed in Section 4 was documented in [I-D.ietf-v6ops-v6onbydefault] by

Sebastien Roy, Alain Durand and James Paugh.

10. References

10.1. 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.
- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2463] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

10.2. Informative References

- [Guynes] Guynes, J., "Impact of System Response Time on State Anxiety", Communications of the ACM , 1988.
- [I-D.gont-tcpm-icmp-attacks]
Gont, F., "ICMP attacks against TCP",
draft-gont-tcpm-icmp-attacks-05 (work in progress),
October 2005.
- [I-D.ietf-v6ops-v6onbydefault]
Roy, S., Durand, A., and J. Paugh, "Issues with Dual Stack IPv6 on by Default", draft-ietf-v6ops-v6onbydefault-03 (work in progress), July 2004.
- [Linux] The Linux Project, "<http://www.kernel.org>".
- [RFC0816] Clark, D., "Fault isolation and recovery", RFC 816, July 1982.
- [Shneiderman]
Shneiderman, B., "Response Time and Display Rate in Human

Performance with Computers", ACM Computing Surveys , 1984.

[Stevens] "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley , 1994.

[Stevens2]

Wright, G. and W. Stevens, "TCP/IP Illustrated, Volume 2: The Implementation", Addison-Wesley , 1994.

[Thadani] Thadani, A., "Interactive User Productivity", IBM Systems Journal No. 1, 1981.

Appendix A. Other possible solutions

A.1. A more conservative approach

A more conservative approach would be to abort a connection in the SYN-SENT or SYN-RECEIVED states only after an ICMP Destination Unreachable has been received a specified number of times, and the SYN segment has been retransmitted more than some specified number of times.

Two new parameters would have to be introduced to TCP, to be used only during the connection-establishment phase: MAXSYNREXMIT and MAXSOFTERROR. MAXSYNREXMIT would specify the number of times the SYN segment would have to be retransmitted before a connection is aborted. MAXSOFTERROR would specify the number of ICMP messages indicating soft errors that would have to be received before a connection is aborted.

Two additional variables would need to be introduced to store additional state information during the connection-establishment phase: "nsynrexmit" and "nsofterror". Both would be initialized to zero. "nsynrexmit" would be incremented by one every time the SYN segment is retransmitted. "nsofterror" would be incremented by one every time an ICMP message that indicates a soft error is received.

A connection in the SYN-SENT or SYN-RECEIVED states would be aborted if nsynrexmit was greater than MAXSYNREXMIT and "nsofterror" was simultaneously greater than MAXSOFTERROR.

This approach would give the network more time to solve the connectivity problem. However, it should be noted that depending on the values chosen for the MAXSYNREXMIT and MAXSOFTERROR parameters, this approach could still lead to long delays between connection establishment attempts, thus not solving the problem. For example, BSD systems abort connections in the SYN-SENT or the SYN-RECEIVED

state when a second ICMP error is received, and the SYN segment has been retransmitted more than three times. They also set up a "connection-establishment timer" that imposes an upper limit on the time the connection establishment attempt has to succeed, which expires after 75 seconds [Stevens2]. Even when this policy may be better than the three-minute timeout policy specified in [RFC1122], it may still be inappropriate for handling the potential problems described in this document. This more conservative approach has been implemented in BSD systems since, at least, 1994 [Stevens2].

A.2. Asynchronous Application Notification

In section 4.2.4.1, [RFC1122] states that there MUST be a mechanism for reporting soft TCP error conditions to the application. Such a mechanism (assuming one is implemented) could be used by applications to cycle through the destination IP addresses. However, this approach would increase application complexity, and would take a long time to kick in, as it would require all existing applications to be modified.

A.3. Issuing several connection requests in parallel

For those scenarios in which a domain name maps to several IP addresses, several connection requests could be issued in parallel, each one to a different destination IP address. The host would then use the first connection attempt to succeed, eliminating the potential delay in establishing a connection with the destination host. However, this would mean that every attempt to connect to a multihomed host would imply sending several SYN segments, making it hard for network operators to distinguish valid connection attempts from those performing Denial of Service (DoS) attacks.

An alternative approach would be as follows. A host would issue a connection request to the first IP address in the list returned by the name-to-address mapping function. If this connection request didn't succeed in some time, a connection request to the second IP address in the list would be issued in parallel. If none of these connection requests succeeded in some time, and there were still more addresses left in the list, they would be tried in the same way. While this approach would, in principle, avoid the problems of the previous approach, it might be hard to define the time interval to wait before issuing each parallel connection request. A short time interval would lead to the problems caused by the previous approach, while a long time interval would likely still lead to long delays in establishing a connection with the destination host.

In any case, it must be noted that both approaches have the same drawbacks as the solution described in Appendix A.2: they would

increase application complexity, and would take too long to begin to be used by applications.

Appendix B. Change log (to be removed before publication of the document as an RFC)

B.1. Changes from draft-gont-tcpm-tcp-soft-errors-02

- o Draft resubmitted as draft-ietf.
- o Miscellaneous editorial changes

B.2. Changes from draft-gont-tcpm-tcp-soft-errors-01

- o Changed wording to describe the mechanism, rather than proposing it
- o Miscellaneous editorial changes

B.3. Changes from draft-gont-tcpm-tcp-soft-errors-00

- o Added reference to the Linux implementation in Section 4
- o Added Section 5
- o Added Section 6
- o Added Appendix A.1
- o Moved section "Asynchronous Application Notification" to Appendix A.2
- o Added a Appendix A.3
- o Miscellaneous editorial changes

Author's Address

Fernando Gont
Universidad Tecnologica Nacional
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fernando@gont.com.ar
URI: <http://www.gont.com.ar>

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

