

TCP Maintenance and Minor
Extensions (tcpm)
Internet-Draft
Intended status: Standards Track
Expires: May 22, 2008

L. Eggert
Nokia
F. Gont
UTN/FRH
November 19, 2007

TCP User Timeout Option
draft-ietf-tcpm-tcp-uto-08

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 May 22, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

The TCP user timeout controls how long transmitted data may remain unacknowledged before a connection is forcefully closed. It is a local, per-connection parameter. This document specifies a new TCP option - the TCP User Timeout Option - that allows one end of a TCP connection to advertise its current user timeout value. This information provides advice to the other end of the TCP connection to adapt its user timeout accordingly. Increasing the user timeouts on

Internet-Draft

TCP User Timeout Option

November 2007

both ends of a TCP connection allows it to survive extended periods without end-to-end connectivity. Decreasing the user timeouts allows busy servers to explicitly notify their clients that they will maintain the connection state only for a short time without connectivity.

Table of Contents

1.	Introduction	3
2.	Conventions	4
3.	Operation	4
3.1.	Changing the Local User Timeout	6
3.2.	UTO Option Reliability	8
3.3.	Option Format	9
3.4.	Reserved Option Values	10
4.	Interoperability Issues	10
4.1.	Middleboxes	10
4.2.	TCP Keep-Alives	10
5.	Security Considerations	11
6.	IANA Considerations	12
7.	Acknowledgments	12
8.	References	12
8.1.	Normative References	12
8.2.	Informative References	13
Appendix A.	Document Revision History	14
	Authors' Addresses	15
	Intellectual Property and Copyright Statements	17

1. Introduction

The Transmission Control Protocol (TCP) specification [[RFC0793](#)] defines a local, per-connection "user timeout" parameter that specifies the maximum amount of time that transmitted data may remain unacknowledged before TCP will forcefully close the corresponding connection. Applications can set and change this parameter with OPEN and SEND calls. If an end-to-end connectivity disruption lasts longer than the user timeout, no acknowledgments will be received for any transmission attempt, including keep-alives, and the TCP connection will close when the user timeout occurs.

This document specifies a new TCP option - the TCP User Timeout Option - that allows one end of a TCP connection to advertise its current user timeout value. This information provides advice to the other end of the connection to adapt its user timeout accordingly. That is, TCP remains free to disregard the advice provided by the UTO option if local policies suggest it to be appropriate.

Increasing the user timeouts on both ends of a TCP connection allows it to survive extended periods without end-to-end connectivity. Decreasing the user timeouts allows busy servers to explicitly notify their clients that they will maintain the connection state only for a short time without connectivity.

In the absence of an application-specified user timeout, the TCP specification [[RFC0793](#)] defines a default user timeout of 5 minutes. The Host Requirements RFC [[RFC1122](#)] refines this definition by introducing two thresholds, R1 and R2 ($R2 > R1$), that control the number of retransmission attempts for a single segment. It suggests that TCP should notify applications when R1 is reached for a segment, and close the connection when R2 is reached. [[RFC1122](#)] also defines the recommended values for R1 (three retransmissions) and R2 (100 seconds), noting that R2 for SYN segments should be at least 3 minutes. Instead of a single user timeout, some TCP implementations offer finer-grained policies. For example, Solaris supports

different timeouts depending on whether a TCP connection is in the SYN-SENT, SYN-RECEIVED, or ESTABLISHED state [[SOLARIS-MANUAL](#)].

Although some TCP implementations allow applications to set their local user timeout, TCP has no in-protocol mechanism to signal changes to the local user timeout to the other end of a connection. This causes local changes to be ineffective in allowing a connection to survive extended periods without connectivity, because the other end will still close the connection after its user timeout expires.

The ability to inform the other end of a connection about the local user timeout can improve TCP operation in scenarios that are

currently not well supported. One example of such a scenario is mobile hosts that change network attachment points based on current location. Such hosts, maybe using Mobile IP [[RFC3344](#)], HIP [[RFC4423](#)] or transport-layer mobility mechanisms [[I-D.eddy-tcp-mobility](#)], are only intermittently connected to the Internet. In between connected periods, mobile hosts may experience periods without end-to-end connectivity. Other factors that can cause transient connectivity disruptions are high levels of congestion or link or routing failures inside the network. In these scenarios, a host may not know exactly when or for how long connectivity disruptions will occur, but it might be able to determine an increased likelihood for such events based on past mobility patterns and thus benefit from using longer user timeouts. In other scenarios, the time and duration of a connectivity disruption may even be predictable. For example, an orbiting node on a non-geostationary satellite might experience connectivity disruptions due to line-of-sight blocking by other planetary bodies. The timing of these events may be computable from orbital mechanics.

[2.](#) Conventions

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](#)].

[3.](#) Operation

Use of the TCP User Timeout Option can be enabled either on a per-connection basis, e.g., through a socket option, or controlled by a system-wide setting. TCP maintains four per-connection state variables to control the operation of the UTO option, three of which (ADV_UTO, ENABLED and CHANGEABLE) are new:

USER_TIMEOUT

TCP's USER TIMEOUT parameter, as specified in [[RFC0793](#)].

ADV_UTO

UTO option advertised to the remote TCP peer. This is an application-specified value, and may be specified on a system-wide basis. If unspecified, it defaults to the default system-wide USER TIMEOUT.

ENABLED (Boolean)

Flag that controls whether the UTO option is enabled for a connection. Defaults to false.

CHANGEABLE (Boolean)

Flag that controls whether USER_TIMEOUT (TCP's USER TIMEOUT parameter) may be changed based on an UTO option received from the other end of the connection. Defaults to true and becomes false when an application explicitly sets USER_TIMEOUT.

Note that an exchange of UTO options between both ends of a connection is not a binding negotiation. Transmission of a UTO option is a suggestion that the other end consider adapting its user timeout. This adaptation only happens if the the other end of the connection has explicitly allowed it (both ENABLED and CHANGEABLE are true).

Before opening a connection, an application that wishes to use the UTO option enables its use by setting ENABLED to true. It may choose an appropriate local UTO by explicitly setting ADV_UTO; otherwise, UTO is set to the default USER TIMEOUT value. Finally, the application should determine whether it will allow the local USER TIMEOUT to change based on received UTO options from the other end of a connection. The default is to allow this for connections that do not have specific user timeout concerns. If an application explicitly sets the USER_TIMEOUT, CHANGEABLE MUST become false, to

prevent UTO options from the other end to override local application requests. Alternatively, applications can set or clear CHANGEABLE directly through socket API calls.

Performing these steps before an active or passive open causes UTO options to be exchanged in the SYN and SYN-ACK packets and is a reliable way to initially exchange, and potentially adapt to, UTO values. TCP implementations MAY provide system-wide default settings for the ENABLED, ADV_UTO and CHANGEABLE connection parameters.

In addition to exchanging UTO options in the SYN segments, a connection that has enabled UTO options SHOULD include a UTO option in the first packet that does not have the SYN flag set. This helps to minimize the amount of state information TCP must keep for connections in non-synchronized states, and is particularly useful when mechanisms such as "SYN cookies" [[RFC4987](#)] are implemented, allowing a newly-established TCP connection to benefit from the information advertised by the UTO option, even if the UTO contained in the initial SYN segment was not recorded.

A host that supports the UTO option SHOULD include one in the next possible outgoing segment whenever it starts using a new user timeout for the connection. This allows the other end of the connection to adapt its local user timeout accordingly. A TCP implementation that does not support the UTO option MUST silently ignore it [[RFC1122](#)], thus ensuring interoperability.

Hosts MUST impose upper and lower limits on the user timeouts they use for a connection. [Section 3.1](#) discusses user timeout limits and potentially problematic effects of some user timeout settings.

Finally, it is worth noting that TCP's option space is limited to 40 bytes. As a result, if other TCP options are in use, they may already consume all the available TCP option space, thus preventing the use of the UTO option specified in this document. Therefore, TCP option space issues should be considered before enabling the UTO option.

[3.1](#). Changing the Local User Timeout

When a host receives a TCP User Timeout Option, it must decide whether to change the local user timeout of the corresponding

connection. If the CHANGEABLE flag is false, USER_TIMEOUT MUST NOT be changed, regardless of the received UTO option. Without this restriction, the UTO option would modify TCP semantics, because an application-requested USER TIMEOUT could be overridden by peer requests. In this case TCP SHOULD, however, notify the application about the user timeout value received from the other end system.

In general, unless the application on the local host has requested a specific USER TIMEOUT for the connection, CHANGEABLE will be true and hosts SHOULD adjust the local TCP USER TIMEOUT (USER_TIMEOUT) in response to receiving a UTO option, as described in the remainder of this section.

The UTO option specifies the user timeout in seconds or minutes, rather than in number of retransmissions or round-trip times (RTTs). Thus, the UTO option allows hosts to exchange user timeout values from 1 second to over 9 hours at a granularity of seconds, and from 1 minute to over 22 days at a granularity of minutes.

Very short USER TIMEOUT values can affect TCP transmissions over high-delay paths. If the user timeout occurs before an acknowledgment for an outstanding segment arrives, possibly due to packet loss, the connection closes. Many TCP implementations default to USER TIMEOUT values of a few minutes. Although the UTO option allows suggestion of short timeouts, applications advertising them should consider these effects.

Long USER TIMEOUT values allow hosts to tolerate extended periods without end-to-end connectivity. However, they also require hosts to maintain the TCP state information associated with connections for long periods of time. [Section 5](#) discusses the security implications of long timeout values.

To protect against these effects, implementations MUST impose limits on the user timeout values they accept and use. The remainder of this section describes a RECOMMENDED scheme to limit TCP's USER TIMEOUT based on upper and lower limits.

Under the RECOMMENDED scheme, and when CHANGEABLE is true, each end SHOULD compute the local USER TIMEOUT for a connection according to this formula:

$USER_TIMEOUT = \min(U_LIMIT, \max(ADV_UTO, REMOTE_UTO, L_LIMIT))$

Each field is to be interpreted as follows:

USER_TIMEOUT

USER TIMEOUT value to be adopted by the local TCP for this connection.

U_LIMIT

Current upper limit imposed on the user timeout of a connection by the local host.

ADV_UTO

User timeout advertised to the remote TCP peer in a TCP User Timeout Option.

REMOTE_UTO

Last user timeout value received from the other end in a TCP User Timeout Option.

L_LIMIT

Current lower limit imposed on the user timeout of a connection by the local host.

The RECOMMENDED formula results in the maximum of the two advertised values, adjusted for the configured upper and lower limits, to be adopted for the user timeout of the connection on both ends. The rationale is that choosing the maximum of the two values will let the connection survive longer periods without end-to-end connectivity. If the end that announced the lower of the two user timeout values did so in order to reduce the amount of TCP state information that must be kept on the host, it can close or abort the connection whenever it wants.

It must be noted that the two endpoints of the connection will not necessarily adopt the same user timeout.

Enforcing a lower limit (L_LIMIT) prevents connections from closing due to transient network conditions, including temporary congestion,

mobility hand-offs and routing instabilities.

An upper limit (U_LIMIT) can reduce the effect of resource exhaustion attacks. [Section 5](#) discusses the details of these attacks.

Note that these limits MAY be specified as system-wide constants or at other granularities, such as on per-host, per-user, per-outgoing-interface or even per-connection basis. Furthermore, these limits need not be static. For example, they MAY be a function of system resource utilization or attack status and could be dynamically adapted.

The Host Requirements RFC [[RFC1122](#)] does not impose any limits on the length of the user timeout. However, it recommends a time interval of at least 100 seconds. Consequently, the lower limit (L_LIMIT) SHOULD be set to at least 100 seconds when following the RECOMMENDED scheme described in this section. Adopting a user timeout smaller than the current retransmission timeout (RTO) for the connection would likely cause the connection to be aborted unnecessarily. Therefore, the lower limit (L_LIMIT) MUST be larger than the current retransmission timeout (RTO) for the connection. It is worth noting that an upper limit may be imposed on the RTO, provided it is at least 60 seconds [[RFC2988](#)].

[3.2.](#) UTO Option Reliability

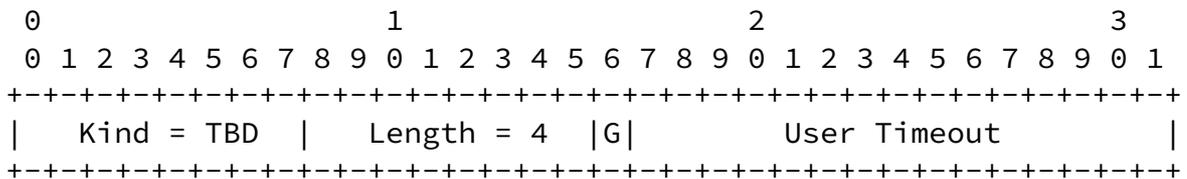
The TCP User Timeout Option is an advisory TCP option that does not change processing of subsequent segments. Unlike other TCP options, it need not be exchanged reliably. Consequently, the specification does not define a reliability handshake for UTO option exchanges. When a segment that carries a UTO option is lost, the other end will simply not have the opportunity to update its local UTO.

Implementations MAY implement local mechanisms to improve delivery reliability, such as retransmitting a UTO option when they retransmit a segment that originally carried it, or "attaching" the option to a byte in the stream and retransmitting the option whenever that byte or its ACK are retransmitted.

It is important to note that although these mechanisms can improve transmission reliability for the UTO option, they do not guarantee delivery (a three-way handshake would be required for this). Consequently, implementations MUST NOT assume that UTO options are transmitted reliably.

3.3. Option Format

Sending a TCP User Timeout Option informs the other end of the connection of the current local user timeout and suggests that the other end adapt its user timeout accordingly. The user timeout value included in a UTO option contains the ADV_UTO value, that is expected to be adopted for the TCP's USER TIMEOUT parameter during the synchronized states of a connection (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, or LAST-ACK). Connections in other states MUST use the default timeout values defined in [RFC0793] and [RFC1122].



(One tick mark represents one bit.)

Figure 1: Format of the TCP User Timeout Option

Figure 1 shows the format of the TCP User Timeout Option. It contains these fields:

Kind (8 bits)

This MUST be TBD, i.e., the TCP option number [RFC0793] assigned by IANA upon publication of this document (see [Section 6](#)). [[Note to the RFC Editor: Throughout this document, replace "TBD" with the TCP option number that IANA has allocated and remove this note.]]

Length (8 bits)

Length of the TCP option in octets [RFC0793]; its value MUST be 4.

Granularity (1 bit)

Granularity bit, indicating the granularity of the "User Timeout" field. When set (G = 1), the time interval in the "User Timeout" field MUST be interpreted as minutes. Otherwise (G = 0), the time interval in the "User Timeout" field MUST be interpreted as seconds.

User Timeout (15 bits)

Specifies the user timeout suggestion for this connection. It MUST be interpreted as a 15-bit unsigned integer. The granularity of the timeout (minutes or seconds) depends on the "G" field.

[3.4.](#) Reserved Option Values

An TCP User Timeout Option with a "User Timeout" field of zero and a "Granularity" bit of either minutes (1) or seconds (0) is reserved for future use. Current TCP implementations MUST NOT send it and MUST ignore it upon reception.

[4.](#) Interoperability Issues

This section discusses interoperability issues related to introducing the TCP User Timeout Option.

[4.1.](#) Middleboxes

A TCP implementation that does not support the TCP User Timeout Option MUST silently ignore it [[RFC1122](#)], thus ensuring interoperability. In a study of the effects of middleboxes on transport protocols, Medina et al. have shown that the vast majority of modern TCP stacks correctly handle unknown TCP options [[MEDINA](#)]. In this study, 3% of connections failed when an unknown TCP option appeared in the middle of a connection. Because the number of failures caused by unknown options is small and they are a result of incorrectly implemented TCP stacks that violate existing requirements to ignore unknown options, they do not warrant special measures. Thus, this document does not define a mechanism to negotiate support of the TCP User Timeout Option during the three-way handshake.

Stateful firewalls usually time out connection state after a period of inactivity. If such a firewall exists along the path, it may close or abort connections regardless of the use of the TCP User Timeout Option. In the future, such firewalls may learn to parse the TCP User Timeout Option in unencrypted TCP segments and adapt connection state management accordingly.

[4.2.](#) TCP Keep-Alives

Some TCP implementations, such as those in BSD systems, use a different abort policy for TCP keep-alives than for user data. Thus,

the TCP keep-alive mechanism might abort a connection that would otherwise have survived the transient period without connectivity. Therefore, if a connection that enables keep-alives is also using the TCP User Timeout Option, then the keep-alive timer MUST be set to a value larger than that of the adopted USER TIMEOUT.

5. Security Considerations

Lengthening user timeouts has obvious security implications. Flooding attacks cause denial of service by forcing servers to commit resources for maintaining the state of throw-away connections. However, TCP implementations do not become more vulnerable to simple SYN flooding by implementing the TCP User Timeout Option, because user timeouts exchanged during the handshake only affect the synchronized states (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK), which simple SYN floods never reach.

However, when an attacker completes the three-way handshakes of its throw-away connections it can amplify the effects of resource exhaustion attacks, because the attacked server must maintain the connection state associated with the throw-away connections for longer durations. Because connection state is kept longer, lower-frequency attack traffic, which may be more difficult to detect, can already exacerbate resource exhaustion.

Several approaches can help mitigate this issue. First, implementations can require prior peer authentication, e.g., using IPsec [[RFC4301](#)] or TCP-MD5 [[RFC2385](#)], before accepting long user timeouts for the peer's connections. Similarly, a host can start to accept long user timeouts for an established connection only after in-band authentication has occurred, for example, after a TLS handshake across the connection has succeeded [[RFC4346](#)]. Although these are arguably the most complete solutions, they depend on external mechanisms to establish a trust relationship.

A second alternative that does not depend on external mechanisms would introduce a per-peer limit on the number of connections that may use increased user timeouts. Several variants of this approach

are possible, such as fixed limits or shortening accepted user timeouts with a rising number of connections. Although this alternative does not eliminate resource exhaustion attacks from a single peer, it can limit their effects. Reducing the number of high-UTO connections a server supports in the face of an attack turns that attack into a denial-of-service attack against the service of high-UTO connections.

Per-peer limits cannot protect against distributed denial of service attacks, where multiple clients coordinate a resource exhaustion attack that uses long user timeouts. To protect against such attacks, TCP implementations could reduce the duration of accepted user timeouts with increasing resource utilization.

TCP implementations under attack may be forced to shed load by resetting established connections. Some load-shedding heuristics,

such as resetting connections with long idle times first, can negatively affect service for intermittently connected, trusted peers that have suggested long user timeouts. On the other hand, resetting connections to untrusted peers that use long user timeouts may be effective. In general, using the peers' level of trust as a parameter during the load-shedding decision process may be useful. Note that if TCP needs to close or abort connections with a long TCP User Timeout Option to shed load, these connections are still no worse off than without the option.

Finally, upper and lower limits on user timeouts, discussed in [Section 3.1](#), can be an effective tool to limit the impact of these sorts of attacks.

6. IANA Considerations

This section is to be interpreted according to [\[I-D.narten-iana-considerations-rfc2434bis\]](#).

This document does not define any new namespaces. It requests that IANA allocate a new 8-bit TCP option number for the UTO option from the registry maintained at <http://www.iana.org/assignments/tcp-parameters>.

7. Acknowledgments

The following people have improved this document through thoughtful suggestions: Mark Allman, Caitlin Bestler, David Borman, Bob Braden, Marcus Brunner, Wesley Eddy, Gorry Fairhurst, Abolade Gbadegesin, Ted Faber, Guillermo Gont, Tom Henderson, Joseph Ishac, Jeremy Harris, Alfred Hoenes, Phil Karn, Michael Kerrisk, Dan Krejsa, Jamshid Mahdavi, Kostas Pentikousis, Juergen Quittek, Anantha Ramaiah, Joe Touch, Stefan Schmid, Simon Schuetz, Tim Shepard and Martin Stiemerling.

8. References

8.1. Normative References

[I-D.narten-iana-considerations-rfc2434bis]

Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [draft-narten-iana-considerations-rfc2434bis-08](#) (work in progress), October 2007.

Eggert & Gont

Expires May 22, 2008

[Page 12]

Internet-Draft

TCP User Timeout Option

November 2007

[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.

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

8.2. Informative References

[I-D.eddy-tcp-mobility]

Eddy, W., "Mobility Support For TCP", [draft-eddy-tcp-mobility-00](#) (work in progress), April 2004.

[MEDINA] Medina, A., Allman, M., and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", Proc. 4th ACM SIGCOMM/USENIX Conference on Internet

Measurement , October 2004.

- [RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", [RFC 2385](#), August 1998.
- [RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", [RFC 2988](#), November 2000.
- [RFC3344] Perkins, C., "IP Mobility Support for IPv4", [RFC 3344](#), August 2002.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), December 2005.
- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", [RFC 4346](#), April 2006.
- [RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", [RFC 4423](#), May 2006.
- [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", [RFC 4987](#), August 2007.
- [SOLARIS-MANUAL] Sun Microsystems, "Solaris Tunable Parameters Reference Manual", Part No. 806-7009-10, 2002.

[Appendix A](#). Document Revision History

[[Note to the RFC Editor: Section to be removed upon publication.]]

Revision	Comments
-08	Addressed additional, minor working group last call comments.
-07	Addressed working group last call comments.

-06	Includes a note on the limited space for TCP options and miscellaneous editorial changes (suggested by Anantha Ramaiah). Includes possible enforcement of per-outgoing-interface limits for the UTO, and miscellaneous editorial changes (suggested by Alfred Hoenes). Includes relevant changes to reflect WG consensus how the local user timeout should be selected (i.e., record both the current user timeout, and the advertised UTO).
-05	Made behavior on when to change/not change the local UTO in response to incoming options consistent through the document. This required some reshuffling of text and also removed the need for the special "don't care" option value.
-04	Clarified the results obtained by Medina et al. Added text to suggest inclusion of the UTO in the first non-SYN segment by the TCP that sent a SYN in response to an active OPEN.
-03	Corrected use of RFC2119 terminology. Clarified how use of the TCP UTO is triggered. Clarified reason for sending a UTO in the SYN and SYN/ACK segments. Removed discussion of the SO_SNDTIMEO and SO_RCVTIMEO socket options. Removed text that suggested that a UTO should be sent upon receipt of an UTO from the other end. Required minimum value for the lower limit of the user timeout. Moved alternative solutions to appendix. Miscellaneous editorial changes.

-02	Corrected terminology by replacing terms like "negotiate", "coordinate", etc. that were left from pre-WG-document times when the UTO was a more formalized exchange instead of the advisory one it is
-----	---

	now. Application-requested UTOs take precedence over ones received from the peer (pointed out by Ted Faber). Added a brief mention of SO_SNDTIMEO and a slightly longer discussion of SO_RCVTIMEO.
-01	Clarified and corrected the description of the existing user timeout in RFC793 and RFC1122 . Removed distinction between operating during the 3WHS and the established states and introduced zero-second "don't care" UTOs in response to mailing list feedback. Updated references and addressed many other comments from the mailing list.
-00	Resubmission of draft-eggert-gont-tcpm-tcp-uto-option-01.txt to the secretariat after WG adoption. Thus, permit derivative works. Updated Lars Eggert's funding attribution. Updated several references. No technical changes.

Authors' Addresses

Lars Eggert
Nokia Research Center
P.O. Box 407
Nokia Group 00045
Finland

Phone: +358 50 48 24461

Email: lars.eggert@nokia.com

URI: http://research.nokia.com/people/lars_eggert/

Fernando Gont
Universidad Tecnológica Nacional / Facultad Regional Haedo
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/>

Internet-Draft

TCP User Timeout Option

November 2007

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

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.

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, THE IETF TRUST 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.

Intellectual Property

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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).