TCP Maintenance and Minor Extensions (tcpm)
Internet-Draft
Expires: January 16, 2006

L. Eggert
NEC
F. Gont
UTN/FRH
July 15, 2005

TCP User Timeout Option
draft-ietf-tcpm-tcp-uto-01

Status of this Memo

Copyright Notice

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. The advisory TCP User Timeout Option allows conforming TCP implementations to exchange their local user timeouts. This exchange provides an in-protocol mechanism to coordinate raising or lowering the two user timeouts of a connection. Increase the user timeouts allows established TCP connections to

survive extended periods of disconnection. Decreasing user timeouts allows busy servers to explicitly notify their clients that they will maintain the connection state only across short periods of disconnection.

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 a network disconnection lasts longer than the user timeout, no acknowledgments will be received for any transmission attempt, including keep-alives [TCP-ILLU], and the TCP connection will close when the user timeout occurs. 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), on the number of retransmissions of a single segment. It suggests that TCP notify applications when R1 is reached for a segment, and close the connection once R2 is reached. [RFC1122] also refines 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 applications may set their local user timeout, there is no in-protocol mechanism to signal changes in the local user timeout to remote peers. This causes local changes to be ineffective, because, for example, the peer will still close the connection after its user timeout expires, even when a host has raised its local user timeout. The ability to modify the two user timeouts associated with a connection in a coordinated manner can improve TCP operation in scenarios that are currently not well supported. One example of such scenarios are mobile hosts that change network attachment points based on current location. Such hosts, maybe using MobileIP [RFC3344], HIP [I-D.ietf-hip-arch] 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 of disconnection during which no network service is available [SCHUETZ-THESIS][SCHUETZ-CCR][DRIVE-THRU]. Other factors that can cause transient periods of disconnection are high levels of congestion as well as link or routing failures inside the

network.

In scenarios similar to the ones described above, a host may not know exactly when or for how long it will be disconnected from the network, but it might expect such events due to past mobility patterns and thus benefit from using longer user timeouts. In other scenarios, the length and time of a network disconnection may even be predictable. For example, an orbiting node on a satellite might experience disconnections due to line-of-sight blocking by other planetary bodies. The disconnection periods of such a node may be easily computable from orbital mechanics.

This document specifies a new TCP option - the User Timeout Option (UTO) - that allows conforming hosts to exchange their local, per-connection user timeout information. This allows, for example, mobile hosts to maintain TCP connections across disconnected periods that are longer than their peer's default user timeout. A second use of the TCP User Timeout Option is advertisement of shorter-than-default user timeouts. This can allow busy servers to explicitly notify their clients that they will maintain the state associated with established connections only across short periods of disconnection.

The same benefits can be obtained through an application-layer mechanism, i.e., coordinating changes to the user timeout values of a connection through application messages. This approach does not require a new TCP option, but requires application changes.

A different approach to tolerate longer periods of disconnection is simply increasing the system-wide user timeout on both peers. This approach has the benefit of not requiring a new TCP option. However, it can also significantly increase the amount of connection state information a busy server must maintain, because a longer global timeout value will apply to all its connections. The proposed TCP User Timeout Option, on the other hand, allows hosts to selectively manage the user timeouts of individual connections, reducing the amount of state they must maintain across disconnected periods.

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

Sending a TCP User Timeout Option suggests that the remote peer SHOULD start using the indicated user timeout value for the

corresponding connection. The user timeout value included in a TCP User Timeout Option specifies the requested user timeout 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 standard timeout values [RFC0793][RFC1122]. [anchor4]

Note that an exchange of TCP User Timeout Options between peers is not a binding negotiation. Transmission of a TCP User Timeout Option is an advisory suggestion that the peer consider adapting its local user timeout. Hosts remain free to forcefully close or abort connections at any time for any reason, whether or not they use custom user timeouts or have suggested to the peer to use them.

A host that supports the TCP User Timeout Option SHOULD include it in the next possible segment to its peer whenever it starts using a new user timeout for the connection. This allows the peer to adapt its local user timeout for the connection accordingly.

When a host that supports the TCP User Timeout Option receives one, it decides whether to change its local user timeout of the connection based on the received value. Generally, hosts should honor requests for changes to the user timeout (see Section 3.3), unless security concerns, resource constraints or external policies indicate otherwise (see Section 5). If so, hosts may ignore incoming TCP User Timeout Options and use a different user timeout for the connection.

When a host receives a TCP User Timeout Option, it first decides whether to change its local user timeout for the connection (see Section 3.3) and then decides whether to send a TCP User Timeout Option to its peer in response. If it has never sent a TCP User Timeout Option to its peer during the lifetime of the connection or if it has changed its local user timeout, it SHOULD send TCP User Timeout Option with its current local user timeout to its peer. [anchor5]

A host that supports the TCP User Timeout Option SHOULD include one in each packet that carries a SYN flag, but need not. [MEDINA] has shown that unknown options are correctly handled by the vast majority of modern TCP stacks. It is thus not necessary to require negotiation use of the TCP User Timeout Option for a connection.

A TCP implementation that does not support the TCP User Timeout Option MUST silently ignore it [RFC1122], thus ensuring interoperability.

Hosts SHOULD impose upper and lower limits on the user timeouts they use. Section 3.3 discusses user timeout limits. A TCP User Timeout Option with a value of zero (i.e., "now") is nonsensical and is used

for a special purpose, see Section 3.4. Section 3.3 discusses potentially problematic effects of other user timeout durations.

3.1 Reliability Considerations

The TCP User Timeout Option is an advisory TCP option that does not change processing for subsequent segments. Unlike other TCP options, it need not be exchanged reliably. Consequently, the specification in this section does not define a reliability handshake for TCP User Timeout Option exchanges. When a segment that carries a TCP User Timeout Option is lost, the option may never reach the intended peer.

Implementations MAY implement local mechanisms to improve delivery reliability, such as retransmitting the TCP User Timeout Option when they retransmit the 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 TCP User Timeout Option, they do not guarantee delivery (a three-way handshake would be required for this). Consequently, implementations MUST NOT assume that a TCP User Timeout Option is reliably transmitted.

3.2 Option Format

```
      0                   1                   2                     3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |     Kind = X  |   Length = 4  |G|        User Timeout         |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

(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)
A TCP option number [RFC0793] to be assigned by IANA upon publication of this document (see Section 6).

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.

3.3 Duration of the User Timeout

The TCP User Timeout 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. (An option value of zero is reserved for a special purpose, see Section 3.4.)

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 [TCP-ILLU]. Although the TCP User Timeout Option allows suggestion of short timeouts, applications advertising them SHOULD consider these effects.

Long user timeout values allow hosts to tolerate extended periods of disconnection. 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 SHOULD impose limits on the user timeout values they accept and use. The remainder of this section describes a RECOMMENDED scheme to limit user timeouts based on upper and lower limits. Under the RECOMMENDED scheme, each TCP SHOULD compute the user timeout (USER_TIMEOUT) for a connection according to this formula:

USER_TIMEOUT = min(U_LIMIT, max(LOCAL_UTO, REMOTE_UTO, L_LIMIT))

Each field is to be interpreted as follows:

USER_TIMEOUT
Resulting user timeout value to be adopted by the local TCP for a connection.

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

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

LOCAL_UTO
Current local user timeout of the specific connection.

REMOTE_UTO
Last "user timeout" value suggested by the remote peer by means of the TCP User Timeout Option.

This means that the maximum of the two announced values will be adopted for the user timeout of the connection. The rationale is that choosing the maximum of the two values will let the connection survive longer periods of disconnection. If the TCP 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, nevertheless, close or abort the connection whenever it wants.

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 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, a time interval of at least 100 seconds is RECOMMENDED. Consequently, the lower limit (L_LIMIT) SHOULD be set to at least 100 seconds when following the RECOMMENDED scheme described in this section.

3.4 Special Option Values

Whenever it is legal to do so according to the specification in the previous sections, TCP implementations MAY send a zero-second TCP User Timeout Option, i.e, with a "User Timeout" field of zero and a "Granularity" of zero. This signals their peers that they support the option, but do not suggest a specific user timeout value at that time. Essentially, a zero-second TCP User Timeout Option acts as a "don't care" value.

The receiver of a zero-second TCP User Timeout Option SHOULD perform the RECOMMENDED strategy for calculating a new local USER_TIMEOUT described in Section 3.3 with a numeric value of zero seconds for REMOTE_UTO. The sender SHOULD perform the calculation as described in Section 3.3. Essentially, the sender SHOULD adapt the peer's UTO and the receiver SHOULD continue using its local UTO.

A zero-minute TCP User Timeout Option, i.e., with a "User Timeout" field of zero and a "Granularity" bit of one, is reserved for future use. TCP implementations MUST NOT sent 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

The large number of middleboxes (firewalls, proxies, protocol scrubbers, etc.) currently present in the Internet pose some difficulty for deploying new TCP options. Some firewalls may block segments that carry unknown options, preventing connection establishment when the SYN or SYN-ACK contains a TCP User Timeout Option. Some recent results, however, indicate that for new TCP options, this may not be a significant threat, with only 0.2% of web requests failing when carrying an unknown option [MEDINA].

Stateful firewalls usually reset connections after a period of inactivity. If such a firewall exists along the path between two peers, 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 and modify their behavior or the option accordingly.

4.2 TCP Keep-Alives

Some TCP implementations, such as the one 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 of disconnection. Therefore, if a TCP peer enables TCP keep-alives for a connection that is 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. TCP implementations do not become more vulnerable to simple SYN flooding by implementing the TCP User Timeout Option, because user timeouts negotiated 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 cause resource exhaustion.

Several approaches can help mitigate this issue. First, implementations can require prior peer authentication, e.g., using IPsec [I-D.ietf-ipsec-rfc2401bis], before accepting long user timeouts for the peer's connections. Similarly, a host can only start to accept long user timeouts for an established connection after in-band authentication has occurred, for example, after a TLS handshake across the connection has succeeded [RFC2246]. 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 3.3, 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 [RFC2434].

This document does not define any new namespaces. It uses an 8-bit TCP option number maintained by IANA at http://www.iana.org/assignments/tcp-parameters.

7. Acknowledgments

The following people have improved this document through thoughtful suggestions: Mark Allmann, David Borman, Marcus Brunner, Wesley Eddy, Ted Faber, Guillermo Gont, Tom Henderson, Joseph Ishac, Jeremy Harris, Phil Karn, Michael Kerrisk, Dan Krejsa, Kostas Pentikousis, Juergen Quittek, Joe Touch, Stefan Schmid, Simon Schuetz and Martin Stiemerling.

Lars Eggert is partly funded by Ambient Networks, a research project supported by the European Commission under its Sixth Framework Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Ambient Networks project or the European Commission.

8. References

8.1 Normative References

[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.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

8.2 Informative References

[DRIVE-THRU]
Ott, J. and D. Kutscher, "Drive-Thru Internet: IEEE 802.11b for Automobile Users", Proc. Infocom , March 2004.

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

[I-D.ietf-hip-arch]
Moskowitz, R., "Host Identity Protocol Architecture", draft-ietf-hip-arch-02 (work in progress), January 2005.

[I-D.ietf-ipsec-rfc2401bis]
Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", draft-ietf-ipsec-rfc2401bis-06 (work in progress), April 2005.

[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.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC3344] Perkins, C., "IP Mobility Support for IPv4", RFC 3344, August 2002.

[SCHUETZ-CCR]
Schuetz, S., Eggert, L., Schmid, S., and M. Brunner, "Protocol Enhancements for Intermittently Connected

Hosts", To appear: ACM Computer Communication Review, Vol. 35, No. 3, July 2005.

[SCHUETZ-THESIS]
Schuetz, S., "Network Support for Intermittently Connected Mobile Nodes", Diploma Thesis, University of Mannheim, Germany, June 2004.

[SOLARIS-MANUAL]
Sun Microsystems, "Solaris Tunable Parameters Reference Manual", Part No. 806-7009-10, 2002.

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

Editorial Comments

[anchor4] LE: A future version of this document may extend per-connection user timeouts to the SYN-SENT and SYN-RECEIVED states in a way that conforms to the required minimum timeouts.

[anchor5] LE: Should it really always send UTO when it changes the local timeout? I can imagine some ping-pong effect when two hosts user different UTO adoption strategies. But maybe that's OK?

Authors' Addresses

Lars Eggert
NEC Network Laboratories
Kurfuerstenanlage 36
Heidelberg 69115
Germany

Phone: +49 6221 90511 43
Fax: +49 6221 90511 55
Email: lars.eggert@netlab.nec.de
URI: http://www.netlab.nec.de/

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/

Appendix A. Document Revision History

To be removed upon publication

Revision	Comments
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.
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.

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

Acknowledgment

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