TCP Maintenance and Minor
Extensions (tcpm)
Internet-Draft
Expires: April 22, 2005

L. Eggert
NEC
F. Gont
UTN/FRH

October 22, 2004

TCP User Timeout Option
draft-eggert-gont-tcpm-tcp-uto-option-01

Status of this Memo

Copyright Notice

Abstract

The TCP user timeout controls how long transmitted data may remain unacknowledged before a connection is aborted. TCP implementations typically use a single, system-wide user timeout value. The TCP User Timeout Option allows conforming TCP implementations to exchange

requests for individual, per-connection user timeouts. Lengthening the system-wide default user timeout allows established TCP connections to survive extended periods of disconnection. On the other hand, shortening the default user timeout allows busy servers to explicitly notify their clients they will maintain the connection state information only accross short periods of disconnection.

1. Introduction

The Transmission Control Protocol (TCP) specification [1] defines a "user timeout" parameter that specifies the maximum amount of time that transmitted data may remain unacknowledged before TCP will abort the corresponding connection. If a disconnection lasts longer than the user timeout, no acknowledgments will be received for any transmission attempt, including keep-alives [5], and the TCP connection will be aborted when the user timeout occurs.

The TCP specification [1] does not constrain the permitted values for user timeouts. However, the Host Requirements RFC [2] mandates a timeout of at least three minutes for the SYN-SENT case. Many TCP implementations default to user timeout values of a few minutes [5]. 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 [6].

System-wide user timeouts are a useful basic policy. However, the ability to selectively choose individual user timeout values for different connections 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 [7], HIP [8] or transport-layer mobility mechanisms [9], 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 [10][11][12]. 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.

In the examples above, as well as in other cases, established TCP connections between two peers may be aborted if a disconnection exceeds the system-wide default user timeout. This document specifies a new TCP option - the User Timeout Option - that allows conforming hosts to exchange per-connection user timeout requests. This allows, for example, mobile hosts to maintain TCP connections across disconnected periods that are longer than their system'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.

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 host must maintain, because a longer global timeout value will apply to all its connections. The proposed User Timeout Option, on the other hand, allows hosts to selectively manage the user timeouts of individual connections. They must then only maintain the state associated with selected connections across disconnected periods.

A second benefit of the TCP User Timeout Option is that it allows hosts to both request specific user timeouts for new connections and to request changes to the effective user timeouts of established connections. The latter allows connections to start with short timeouts and only request longer timeouts when disconnection is imminent, and only for connections considered important. The ability to request changes to user timeouts of established connections is also useful to raise the user timeout after in-band authentication has occurred. For example, peers could request longer user timeouts for the TCP connections underlying two-way authenticated TLS connections [13] after their authentication handshakes have succeeded.

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 RFC 2119 [3].

3. Operation

Sending a TCP User Timeout Option suggests to the remote peer to use the indicated user timeout value for the corresponding connection. Section 3.4 discusses the effects of different timeout values.

The user timeout value included in a TCP User Timeout Option specifies the requested user timeout during a connection's synchronized states (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, or LAST-ACK.) Connections in other states MUST use standard timeout values [1][2]. [Comment.1]

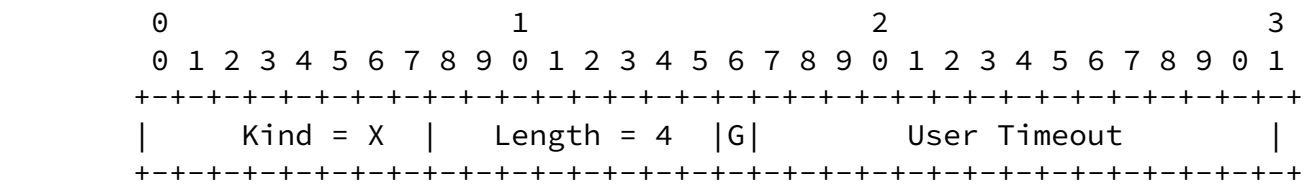
When a host that supports the TCP User Timeout Option receives one, it decides whether to change the connection's local user timeout based on the received value. Generally, hosts SHOULD honor requests for changes to the user timeout, unless security concerns or external policies indicate otherwise (see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

It is important to note that the TCP User Timeout Option does not change the semantics of the TCP protocol. Hosts remain free to abort connections at any time for any reason, whether or not they use custom user timeouts or have suggested the peer to use them.

Hosts SHOULD impose upper and lower limits on the user timeouts they use. Section 3.4 discusses user timeout limits. A TCP User Timeout Option with a value of zero (i.e., "now") is nonsensical and MUST NOT be sent. If received, it MUST be ignored. Section 3.4 discusses potentially problematic effects of other user timeout durations.

A TCP implementation that does not support the TCP User Timeout Option SHOULD silently ignore it [2], thus ensuring interoperability.

3.1 Option Format



(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 [1] to be assigned by IANA upon publication of this document (see Section 6.)

Length (8 bits)
Length of the TCP option in octets [1]; 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.

3.2 Operation During the SYN Handshake

A host that supports the TCP User Timeout Option MUST include an appropriate TCP User Timeout Option in its initial SYN segment to indicate that it supports the option and to suggest an initial user timeout for the connection. [Comment.2]

A host that supports the TCP User Timeout Option and receives a SYN segment that includes one MUST respond with an appropriate TCP User Timeout Option in its SYN-ACK segment. If an incoming SYN segment

does not include a TCP User Timeout Option, a host MUST NOT include one in the SYN-ACK segment nor in any other segment, and it MUST ignore the contents of any other received TCP User Timeout Option.

3.3 Operation During the Synchronized States

Unless both the SYN and SYN-ACK of a connection contained TCP User Timeout Options, both hosts participating in the connection MUST NOT send TCP User Timeout Options in any other segment. Additionally, they both MUST ignore the contents of any received TCP User Timeout Option.

If, however, both the SYN and SYN-ACK contained TCP User Timeout Options, hosts MAY choose to include additional TCP User Timeout Options in segments sent during the synchronized states (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, or LAST-ACK).

Dynamically adapting the user timeout of a connection during its lifetime could be useful in a number of scenarios, for example:

- TCP may adapt the user timeout based on observed network characteristics. [Comment.3]

- TCP may use short timeouts when connections start and only suggest longer timeouts when disconnection was imminent.

- TCP may use short user timeouts when connections start and only raise them once in-band authentication has occurred, for example, once a TLS handshake across the connection has succeeded [13].

Generally, whenever a host decides to change the local user timeout of a connection, it SHOULD include a TCP User Timeout Option indicating the new user timeout in its next segment to the peer.

This allows the peer to adapt its local user timeout for the connection accordingly.

TCP's SYN handshake has specific retransmission rules to guarantee reliability. These mechanisms also guarantee that the exchange of TCP User Timeout Options during the SYN handshake is reliable. This is not the case for TCP User Timeout Option exchanges during the synchronized states. When a segment carrying a TCP User Timeout Option is lost, the peer will not update its local user timeout accordingly. This draft does not currently describe mechanisms to ensure the reliability of the option exchange in the synchronized states, other than noting that periodic inclusion of the option may be an appropriate interim mechanism for implementations concerned with reliability.

3.4 Duration of the User Timeout

The TCP User Timeout Option allows hosts to exchange user timeout values from zero seconds to over 9 hours at a granularity of seconds and from zero minutes 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 aborts. Many TCP implementations default to user timeout values of a few minutes [5]. 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))
[Comment.4]

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 connection's user timeout by the local host.

L_LIMIT
: Current lower limit imposed on the connection's user timeout 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 transient 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, abort the connection whenever it wants.

Enforcing a lower limit (L_LIMIT) protects against connection aborts

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 [2] 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 (LLIMIT) SHOULD be set to at least 100 seconds when following the RECOMMENDED scheme described in this section.

4. Interoperability Issues

This section discusses interoperability issues related to introducing the UTO option.

One meta-issue of introducing new TCP options is that header space available for TCP options is currently limited to 40 bytes. All negotiable options are exchanged during the SYN/SYN-ACK handshake, where option space is becoming limited. Current proposals to extend the available option space may mitigate this issue [14].

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 the UTO 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 [15].

Stateful firewalls usually reset connections after a period of inactivity. If such a firewall exists along the path between two peers, it may abort connections regardless of the use of the UTO Option. In the future, such firewalls may learn to parse the UTO option and modify their behavior accordingly.

4.2 TCP Keep-Alives

Some TCP implementations, such as the one in BSD systems, use a different abort policy for TCP keepalives 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 UTO Option, then the keep-alive timer MUST be set to a value larger than that of the adopted USER TIMEOUT (specified by Equation 1).

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. [Comment.5]

Several approaches can help mitigate this issue. First, implementations can require prior peer authentication, e.g., using IPsec [16], 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 [13]. 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.

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.

Finally, upper and lower limits on user timeouts, discussed in Section 3.4, 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 [4].

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, Phil Karn, Michael Kerrisk, Kostas Pentikousis, Juergen Quittek, Joe Touch, Stefan Schmid, Simon Schuetz and Martin Stiemerling.

Part of this work is a byproduct of the Ambient Networks project, partially supported by the European Commission under its Sixth Framework Programme. It is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranties of fitness for a particular purpose. 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

[1] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[2] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

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

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

8.2 Informative References

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

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

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

[8] Moskowitz, R., "Host Identity Protocol Architecture", draft-moskowitz-hip-arch-06 (work in progress), June 2004.

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

[10] Schuetz, S., "Network Support for Intermittently Connected

Mobile Nodes", M.S. Thesis, University of Mannheim, Germany, June 2004.

[11] Schuetz, S., Eggert, L., Schmid, S. and M. Brunner, "Protocol Enhancements for Intermittently Connected Hosts", under submission (work in progress), July 2004.

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

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

[14] Eddy, W., "Extending the Space Available for TCP Options", draft-eddy-tcp-loo-01 (work in progress), September 2004.

[15] Medina, A., Allman, M. and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", To appear: Proc. ACM SIGCOMM/USENIX Internet Measurement Conference , October 2004.

[16] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

Editorial Comments

[Comment.1] 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.

[Comment.2] LE: My original proposal was to allow hosts to choose

whether or not to include the option. It's open for discussion whether this flexibility is worth the additional complexity. This is the corresponding text: "A host that supports the TCP User Timeout Option MAY omit the TCP User Timeout Option from the initial SYN if it will not permit custom user timeouts for the specific connection. It SHOULD omit the TCP User Timeout Option from the initial SYN if there is evidence that the peer does not support the TCP User Timeout Option, for example, if a prior connection attempt including a TCP User Timeout Option has failed. If a host does not include a TCP User Timeout Option in its initial SYN, it MUST NOT include it in any other segment either and MUST ignore the contents of any received TCP User Timeout Option."

[Comment.3] FG: My original proposal suggested that TCP might adapt the user timeout when signalled of congestion by means of ECN.

[Comment.4] LE: This formula takes the maximum of the two announced values. I'd use USER_TIMEOUT = max(L_LIMIT, min(LOCAL_UTO, REMOTE_UTO, U_LIMIT)), instead. This version takes the minimum. My rationale is that the party announcing the lower value probably had a reason for it and may hence not be prepared to handle a longer value that it originally indicated.

[Comment.5] FG: IMO, in practice the TCP User Timeout option does not make the situation worse: the same type of attack can be performed even if the default "USER TIMEOUT" is used, since TCP requires no message exchange in order to keep a connection open.

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

Revision	Comments
00	Initial version.
01	Merged the ATO and AUTO drafts.

Intellectual Property Statement

ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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