

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 8, 2009

M. Scharf
University of Stuttgart
S. Floyd
ICIR
P. Sarolahti
Nokia Research Center
July 7, 2008

TCP Flow Control for Fast Startup Schemes
draft-scharf-tcpm-flow-control-quick-start-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 January 8, 2009.

Abstract

This document describes extensions for the flow control of the Transmission Control Protocol (TCP) that avoid interactions with fast startup congestion control mechanisms, in particular the Quick-Start TCP extension. Quick-Start is an optional TCP extension that allows to start data transfers with a large congestion window, using feedback of the routers along the path. This can avoid the time consuming Slow-Start, provided that the TCP flow control is not a limiting factor.

There are two potential interactions between the TCP flow control and congestion control schemes without the standard Slow-Start: First, receivers might not allocate a sufficiently large buffer space after connection setup, or they may advertise a receive window implicitly assuming the Slow-Start behavior on the sender side. This document therefore provides guidelines for buffer allocation in hosts supporting the Quick-Start extension. Second, the TCP receive window scaling mechanism can prevent fast startups immediately after the initial three-way handshake connection setup. This document describes a simple solution to overcome this problem.

Table of Contents

1.	Introduction	3
2.	Requirements Notation	4
3.	Receive Buffer Dimensioning	4
3.1.	Background	4
3.2.	Recommendations for Buffer Dimensioning with Quick-Start	4
4.	Receive Window Scaling Issues	5
4.1.	Background	5
4.2.	Interaction Problem	6
4.3.	Proposed Solution	6
4.4.	Deployment Considerations	8
5.	Security Considerations	9
6.	IANA Considerations	9
7.	Acknowledgments	9
8.	References	10
8.1.	Normative References	10
8.2.	Informative References	10
Appendix A.	Applicability to Other Proposals	11
Appendix B.	Alternative Solutions	11
Appendix C.	Document Revision History	12
	Authors' Addresses	12
	Intellectual Property and Copyright Statements	13

1. Introduction

The Transmission Control Protocol (TCP) [[RFC0793](#)] realizes both flow control and congestion control. The TCP flow control is a receiver-driven mechanism that informs the sender about the available receive buffer space and limits the maximum amount of outstanding data. In general, flow control and congestion control are independent mechanisms, and the allocation of receive buffer space is up to the receiving network stack only. But if the TCP connection spans a path with a large bandwidth-delay product (BDP), both congestion and receive window should have large values in order to achieve good TCP performance (see [[RFC2488](#)],[[RFC3481](#)]). This results in some overlap of flow control and congestion control.

A fast startup scheme, which speeds up data transfers by not using the standard Slow-Start mechanism [[RFC2581](#)], can suffer from further interactions between the TCP flow control and congestion control. While not being appropriate for the global Internet, such a fast startup congestion control could be deployed for instance in controlled environments. The experimental Quick-Start TCP extension [[RFC4782](#)] is currently the only specified TCP extension that realizes a fast startup. This is why this document only considers Quick-Start. However, as discussed in [Appendix A](#), interactions between the TCP flow control and congestion control mechanisms could also arise if a fast startup was realized by other means.

With Quick-Start, TCP hosts can request permission from the routers along a network path to send at a higher rate than allowed by the default TCP congestion control, in particular during connection setup or after longer idle periods. The explicit router feedback avoids the time-consuming capacity probing by the TCP Slow-Start and can significantly improve transfer times over paths with a high bandwidth-delay product [[SAF07](#)].

The usage of a fast startup significantly changes the TCP behavior during connection setup, since a sender can use large congestion

windows immediately after connection setup. Concerning the flow control, these large windows raise two questions: First, what receiver buffer allocation strategies should be used? And second, how to appropriately signal large windows? This document addresses these issues and shows that fast startup schemes require special mechanisms in both cases. The document thereby supplements the Quick-Start TCP specification [[RFC4782](#)], where flow control issues have not been addressed in detail.

The rest of this document is structured as follows: First, the question of receive buffer allocation in combination with Quick-Start is addressed and dimensioning guidelines are provided. And second, a

usage of the receive window scaling mechanism [[RFC1323](#)] is specified, which is required to fully benefit from Quick-Start when the Quick-Start request is used in the initial <SYN> segment.

[2.](#) Requirements Notation

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.](#) Receive Buffer Dimensioning

[3.1.](#) Background

According to [[RFC2581](#)], a TCP sender can transmit up to the minimum of the congestion window and the receive window (also called receiver's advertised window). Several factors can have an impact on the value of the receive window: On the one hand, hosts with a potentially high number of TCP connections need to optimize their buffer and memory usage to be able to serve a maximum possible number of TCP connections. On the other hand, a receiver that wants to use the available bandwidth should advertise a receive window that is big enough to allow an efficient utilization of the connection path. Finding a fixed receive buffer size that is optimal between these two goals is difficult.

This is why many modern TCP implementations use an intelligent

dynamic buffer management. There are different auto-tuning techniques and heuristics [[Dun06](#)] designed to prevent the receive window from limiting the data rate at the sender. An implementation using receive window auto-tuning is described for instance in [[SB05](#)]. A common characteristic of most of these buffer allocation strategies is that they initially advertise a rather small receive window. The more data arrives, the more buffer is advertised to the corresponding connection. This behavior is reasonable if the sender uses the standard Slow-Start algorithm and thus starts with a small congestion window anyway. However, when a fast startup shall be used, the receiver must be ready to buffer a large amount of data immediately after the connection setup.

[3.2.](#) Recommendations for Buffer Dimensioning with Quick-Start

A network stack that supports the Quick-Start TCP extension should apply the following guidelines for receive buffer allocation, in addition to the normal buffer management principles:

When a host receives and approves a Quick-Start request, it SHOULD announce a receive window that is large enough so that a potential Quick-Start data transfer can start with a high sending window. If buffer size auto-tuning is used, it SHOULD be ensured that a sufficiently high initial receive window is announced. The handling of buffer space upon arrival of a Quick-Start request SHOULD be configurable by the corresponding application.

The TCP host could estimate the required buffer space as the product of the approved Quick-Start rate and the round-trip time, and advertise a receive window based on this required buffer space. This receive window should allow the other TCP host to fully use the approved Quick-Start Request. If the TCP host doesn't know the round-trip time, the TCP host could use an estimate of the round-trip time in calculating the required buffer space. For instance, the buffer dimension could be done for a configurable "worst-case" RTT such as 500 ms. Alternately, the TCP host could base the advertised receive window on the available buffer space, without calculating the buffer space required for the other TCP host to fully use the approved Quick-Start Request.

[4. Receive Window Scaling Issues](#)

[4.1. Background](#)

The TCP header specified in [[RFC0793](#)] uses a 16 bit field to report the receive window size to the sender. This effectively limits the sending window to 64 KB. To circumvent this limitation, the "Window Scale" TCP extension [[RFC1323](#)] defines an implicit scale factor, which is used to multiply the window size value found in a TCP header to obtain a 32 bit window size. If enabled, the scale factor is announced during connection setup by the "Window Scale" TCP option in <SYN> and <SYN,ACK> segments.

In general, using receive window scaling is highly beneficial for TCP connections over path with a large bandwidth-delay product [[RFC2488](#)],[[RFC3481](#)]. Otherwise, the path capacity cannot fully be utilized by TCP. Quick-Start TCP can significantly speed up data transfers over such paths [[RFC4782](#)],[[SAF07](#)]. As a consequence, a host supporting Quick-Start should enable receive window scaling according to [[RFC1323](#)]. If Quick-Start is used in the initial three-way handshake, the minimum required scaling factor may be obtained from the required receive buffer space, which can be approximated as described in the previous section.

[4.2. Interaction Problem](#)

A problem arises when the Quick-Start mechanism is used within the three-way handshake, and the Quick-Start request is added to the initial <SYN> segment: In this scenario, if the Quick-Start request is approved by the routers along the path, the receiver echoes back the Quick-Start response in the <SYN,ACK> segment. This process is illustrated in [[RFC4782](#)]. Upon reception of the <SYN,ACK> with the Quick-Start response, the sender can set the congestion window to the determined value so that it can immediately start to send with the approved data rate.

However, [[RFC1323](#)] defines that the "Window field in a SYN (i.e., a <SYN> or <SYN,ACK>) segment itself is never scaled." This means that the maximum receive window that can be signaled to the sender in the

<SYN,ACK> is 64 KB. As a consequence, the TCP flow control will prevent the TCP sender from having more than 64 KB of outstanding data, even if the receiver has much more free buffer, and the Quick-Start feedback allows a much larger congestion window.

This effect essentially limits the maximum amount of data sent by Quick-Start to 64 KB, when the sender sends the Quick-Start request in the initial <SYN> segment. Also, the congestion window after quitting the Quick-Start rate pacing phase is at most 64 KB, as the congestion window is set to the amount of data that has actually been sent during the rate pacing phase. This is an undesirable restriction for the Quick-Start mechanism, even if 64 KB is still much more than the initial congestion window in Slow-Start that is allowed by [[RFC3390](#)].

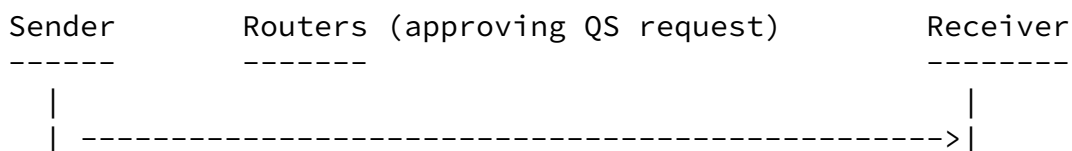
This issue only occurs when Quick-Start is used in the three-way TCP connection setup procedure, and only in the direction of the connection originator to the acceptor. Still, this case is one of the planned usage scenarios for the Quick-Start TCP extension.

[4.3.](#) Proposed Solution

The limitation imposed by the window scaling could be addressed in different ways. This document proposes the following solution: If necessary, the TCP host SHOULD send a scaled receive window in a separate <ACK> packet following the <SYN,ACK> packet.

This means that when a host receives a <SYN> segment with a Quick-Start option, it processes the option as described in [[RFC4782](#)]. Provided that the host has Quick-Start support enabled, the Quick-Start response is echoed back in the <SYN,ACK> segment. As explained, this segment cannot announce receive windows larger than 64 KB. If the receiver allocates a buffer space larger than 64 KB,

an additional empty segment (without <SYN> flag) SHOULD be sent after the <SYN,ACK> segment, in order to announce the true receive window. The resulting message flow is depicted in Figure 1.



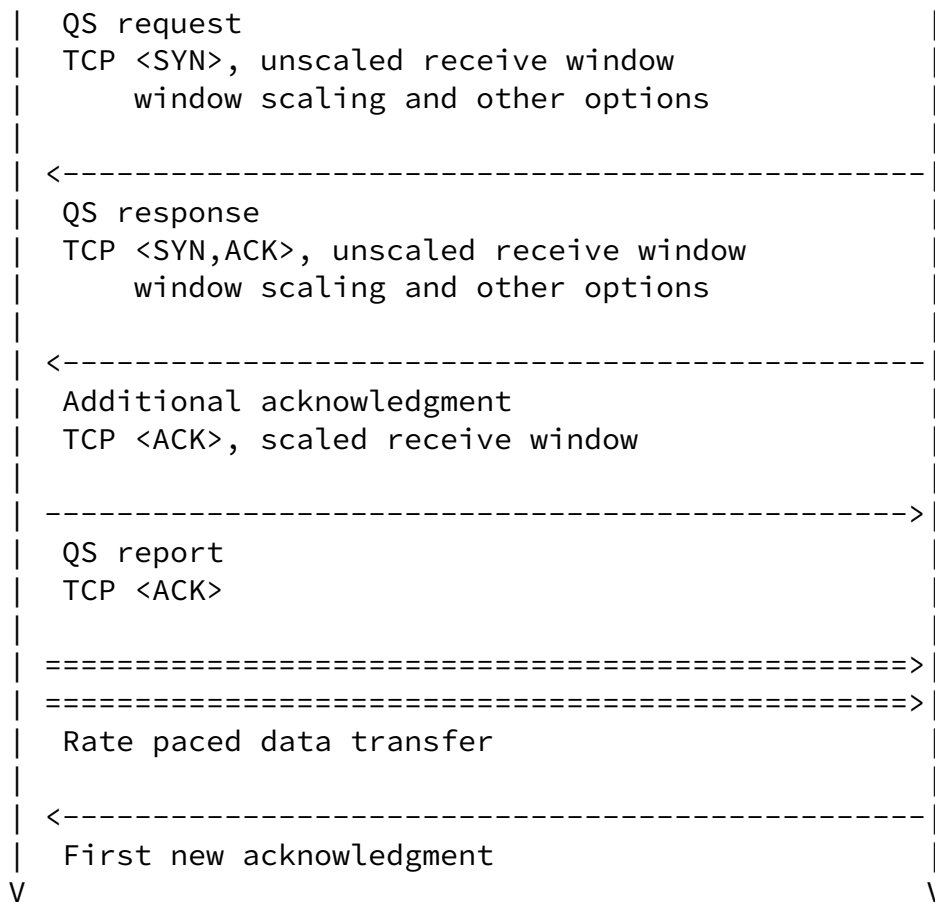


Figure 1: Message sequence chart of the proposed mechanism

After having received this additional acknowledgment, the sender is aware of the true available receive buffer. Provided that the Quick-Start request is approved on the path and that the receive window is sufficiently large, this allows the sender to send more than 64 KB during the Quick-Start rate pacing phase.

We note that there is some degree of freedom as to when to send the additional acknowledgment. The straightforward solution is to send it immediately after the <SYN,ACK> segment. But this is not required: It is sufficient if the sender receives this segment before reaching the limit of the unscaled receive window. As a consequence, receivers could also delay the sending of this segment for some small amount of time.

The method proposed in this document is compliant with the TCP specifications: Sending empty segments to increase the receive window is implicitly allowed by [RFC0793], and in [RFC2581] it is clearly stated that sending an acknowledgment is allowed to update the receive window. For standard-compliant TCP stacks, implementing the method thus should require changes in the receiver TCP implementation only.

However, sending an empty acknowledgment shortly after a <SYN,ACK> segment is an atypical TCP communication event. The <SYN,ACK> and the additional segment could get reordered in the network. In this case, the sending host will typically ignore the additional segment, as it is still awaiting the <SYN,ACK>. Furthermore, middleboxes such as stateful firewalls might drop the additional acknowledgment. Even worse, this segment might also be dropped because a middlebox receives it earlier than the <ACK> segment from the sender. At this point in time, from the viewpoint of the middlebox, the bi-directional end-to-end TCP connection is not yet established. If the additional segment gets dropped, the sender only knows the unscaled receive window until the next new acknowledgment arrives, which may limit the benefit of Quick-Start. Delaying the additional acknowledgment for a short period of time could help to avoid such problems. Further investigation is needed to analyze whether such a delay is required.

A possible alternative to the message flow in Figure 1 would be to piggyback the Quick-Start response on the additional acknowledgment segment instead of the <SYN,ACK>. However, this approach has several drawbacks and is therefore not recommended: First, the Quick-Start response would be received later, which could cause additional delays. Second, the <SYN,ACK> is immediately acknowledged by the <ACK> segment. The Quick-Start rate report can thus be piggybacked on this <ACK>. In contrast, if the Quick-Start response is included in the additional acknowledgment, the Quick-Start report has to be piggybacked to a data segment, i. e., it depends on the availability of application data whether and when the Quick-Start report is sent.

The additional segment mandated by this document results in a network overhead of one segment. In many potential usage scenarios this overhead will be small compared to the network load caused by the acknowledgments of a starting high-speed Quick-Start data transfer.

Instead of sending one additional acknowledgment, a host could also send a small number of copies in order to improve robustness. This could help to reduce the risk of reordering with the <SYN,ACK> segment. However, given the additional overhead, it is recommended

to send only one acknowledgment unless there are indications that the path suffers from frequent packet reordering.

5. Security Considerations

Quick-Start TCP imposes a number of security challenges. Known security threats as well as counter-measures are discussed in the section "Security Considerations" of [\[RFC4782\]](#). Since this document describes extensions to Quick-Start TCP, the security issues and solutions identified in [\[RFC4782\]](#) apply here, too.

If a host reserves large amounts of buffer space during the three-way handshake, this could increase the vulnerability to "syn flooding" attacks: An attacker sending many Quick-Start requests could try to allocate much buffer space at a host, which is then not available any more for other TCP connections. If most involved routers support Quick-Start, this type of attack is difficult to realize, since the routers may reject many requests before they reach a host. However, an attack could be possible if some routers on the path do not support Quick-Start. A simple countermeasure would be to set an upper limit on the total amount of buffer space granted to connections with Quick-Start, and possibly to deny requests if they arrive at a host with too high a frequency. The main impact of this abuse is that Quick-Start may be rendered useless for other connections. This can result in some performance degradation, because the default Slow-Start must be used instead. In general, it is an inherent weak point of Quick-Start that one can send much more requests than required, which temporarily can block resources for other earnest Quick-Start requests [\[RFC4782\]](#).

It is an allowed behavior for a TCP connection endpoint to send an additional acknowledgment segment in order to update the receive window. The usage of the proposed mechanism causes some limited network overhead, but it does not result in additional security threats.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgments

Special thanks to Haiko Strotbek, Martin Koehn, Simon Hauger,

[8.](#) References

[8.1.](#) Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), September 1981.
- [RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", [RFC 1323](#), May 1992.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", [RFC 2581](#), April 1999.
- [RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", [RFC 3390](#), October 2002.
- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", [RFC 4782](#), January 2007.

[8.2.](#) Informative References

- [Dun06] Dunigan, T., "TCP auto-tuning zoo", available at <http://www.csm.ornl.gov/~dunigan/net100/auto.html>, February 2006.
- [FPK07] Falk, A., Pryadkin, Y., and D. Katabi, "Specification for the Explicit Control Protocol (XCP)", Internet Draft, work in progress, June 2007.
- [LAJ+07] Liu, D., Allman, M., Jin, S., and L. Wang, "Congestion Control Without a Startup Phase", Proc. PFLDnet2007, February 2007.
- [RFC2488] Allman, M., Glover, D., and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", [BCP 28](#), [RFC 2488](#), January 1999.

[RFC3481] Inamura, H., Montenegro, G., Ludwig, R., Gurtov, A., and F. Khafizov, "TCP over Second (2.5G) and Third (3G) Generation Wireless Networks", [BCP 71](#), [RFC 3481](#), February 2003.

[SAF07] Sarolahti, P., Allman, M., and S. Floyd, "Determining an Appropriate Sending Rate Over an Underutilized Network Path", *Computer Networks*, vol. 51, no. 7, 2007.

Scharf, et al.

Expires January 8, 2009

[Page 10]

Internet-Draft TCP Flow Control for Fast Startup Schemes

July 2008

[SB05] Smith, M. and S. Bishop, "Flow Control in the Linux Network Stack", available at <http://www.cl.cam.ac.uk/~pes20/Netsem/linuxnet.pdf>, February 2005.

[Appendix A](#). Applicability to Other Proposals

Besides Quick-Start, there are some other fast startup proposals under discussion. A common characteristic is that they can be more aggressive than the standard TCP Slow-Start. A comprehensive survey of this related work can be found in [\[RFC4782\]](#). For instance, the Explicit Control Protocol (XCP) [\[FPK07\]](#) proposes a new congestion control based on explicit router feedback. Furthermore, there are discussions in the research community whether a host could start to send with a large congestion window, combined with a rate pacing mechanism and a conservative reaction in case of congestion [\[LAJ+07\]](#). Basically, the effects discussed in this document are inherent to all fast startup schemes and not specific to Quick-Start.

Dynamic receive buffer dimensioning is a non-trivial task for all fast startup schemes. The amount of information that a receiver can gain during a connection setup procedure differs from proposal to proposal. However, the basic guideline to advertise a larger initial receive window applies to all proposals similar to Quick-Start.

If the TCP header semantics apply, the interaction with receive window scaling mechanism could also be a problem for other approaches. In this case, the workaround of sending an additional acknowledgment can be helpful, too.

[Appendix B.](#) Alternative Solutions

The limitation imposed by the window scaling could be addressed in several ways. This document proposes to send an additional acknowledgment to announce the true receive window, if needed. This method is compliant with the current TCP standards.

Alternatively, one could circumvent [[RFC1323](#)] in several ways. For instance, one could use a scaled receive window in <SYN> and <SYN,ACK> segments, if they include Quick-Start options. The usage of a scaled window could also be indicated by some other means (e. g., a new TCP option). Finally, the advertised window could selectively be ignored by a sender that receives a Quick-Start response. Still, such alternative solutions would require changes in the TCP header semantics and might cause interworking problems with currently deployed TCP implementations.

Scharf, et al.

Expires January 8, 2009

[Page 11]

Internet-Draft TCP Flow Control for Fast Startup Schemes

July 2008

[Appendix C.](#) Document Revision History

This document was originally entitled by "Avoiding Interactions of Quick-Start TCP and Flow Control". Changes from earlier versions of the document include:

- o [draft-scharf-tcpm-flow-control-quick-start-00.txt](#): Changed title and more precise statements on the applicability beyond Quick-Start
- o [draft-scharf-tsvwg-quick-start-flow-control-01.txt](#): Improved description of deployment implications
- o [draft-scharf-tsvwg-quick-start-flow-control-00.txt](#): Initial version

Authors' Addresses

Michael Scharf
University of Stuttgart
Pfaffenwaldring 47
D-70569 Stuttgart
Germany

Phone: +49 711 685 69006

Email: michael.scharf@ikr.uni-stuttgart.de

URI: <http://www.ikr.uni-stuttgart.de/en/~scharf>

Sally Floyd
ICIR (ICSI Center for Internet Research)

Phone: +1 (510) 666-2989
Email: floyd@icir.org
URI: <http://www.icir.org/floyd/>

Pasi Sarolahti
Nokia Research Center
P.O. Box 407
FI-00045 NOKIA GROUP
Finland

Phone: +358 50 4876607
Email: pasi.sarolahti@iki.fi

Scharf, et al.

Expires January 8, 2009

[Page 12]

Internet-Draft TCP Flow Control for Fast Startup Schemes

July 2008

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

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.