Network Working Group
Request for Comments: 4782
Category: Experimental

S. Floyd
M. Allman
ICIR
A. Jain
F5 Networks
P. Sarolahti
Nokia Research Center

January 2007

Quick-Start for TCP and IP

Status of This Memo

Copyright Notice

Abstract

This document specifies an optional Quick-Start mechanism for transport protocols, in cooperation with routers, to determine an allowed sending rate at the start and, at times, in the middle of a data transfer (e.g., after an idle period). While Quick-Start is designed to be used by a range of transport protocols, in this document we only specify its use with TCP. Quick-Start is designed to allow connections to use higher sending rates when there is significant unused bandwidth along the path, and the sender and all of the routers along the path approve the Quick-Start Request.

This document describes many paths where Quick-Start Requests would not be approved. These paths include all paths containing routers, IP tunnels, MPLS paths, and the like that do not support Quick-Start. These paths also include paths with routers or middleboxes that drop packets containing IP options. Quick-Start Requests could be difficult to approve over paths that include multi-access layer-two networks. This document also describes environments where the Quick-Start process could fail with false positives, with the sender incorrectly assuming that the Quick-Start Request had been approved by all of the routers along the path. As a result of these concerns, and as a result of the difficulties and seeming absence of motivation for routers, such as core routers to deploy Quick-Start, Quick-Start is being proposed as a mechanism that could be of use in controlled

environments, and not as a mechanism that would be intended or appropriate for ubiquitous deployment in the global Internet.

Table of Contents

1. Introduction

Each connection begins with a question: "What is the appropriate sending rate for the current network path?" The question is not answered explicitly, but each TCP connection determines the sending rate by probing the network path and altering the congestion window (cwnd) based on perceived congestion. Each TCP connection starts with a pre-configured initial congestion window (ICW). Currently, TCP allows an initial window of between one and four segments of maximum segment size (MSS) ([RFC2581], [RFC3390]). The TCP connection then probes the network for available bandwidth using the slow-start procedure ([Jac88], [RFC2581]), doubling cwnd during each congestion-free round-trip time (RTT).

The slow-start algorithm can be time-consuming --- especially over networks with large bandwidth or long delays. It may take a number of RTTs in slow-start before the TCP connection begins to fully use the available bandwidth of the network. For instance, it takes log_2(N) - 2 round-trip times to build cwnd up to N segments, assuming an initial congestion window of 4 segments. This time in slow-start is not a problem for large file transfers, where the slow-start stage is only a fraction of the total transfer time. However, in the case of moderate-sized transfers, the connection might carry out its entire transfer in the slow-start phase, taking many round-trip times, where one or two RTTs might have been sufficient when using the currently available bandwidth along the path.

A fair amount of work has already been done to address the issue of choosing the initial congestion window for TCP, with RFC 3390 allowing an initial window of up to four segments based on the MSS used by the connection [RFC3390]. Our underlying premise is that explicit feedback from all the routers along the path would be required, in the current architecture, for best-effort connections to use initial windows significantly larger than those allowed by [RFC3390], in the absence of other information about the path.

In using Quick-Start, a TCP host (say, host A) would indicate its desired sending rate in bytes per second, using a Quick-Start Option in the IP header of a TCP packet. Each router along the path could, in turn, either approve the requested rate, reduce the requested rate, or indicate that the Quick-Start Request is not approved. (We note that the 'routers' referred to in this document also include the IP-layer processing of the Quick-Start Request at the sender.) In approving a Quick-Start Request, a router does not give preferential treatment to subsequent packets from that connection; the router is only asserting that it is currently underutilized and believes there is sufficient available bandwidth to accommodate the sender's requested rate. The Quick-Start mechanism can determine if there are routers along the path that do not understand the Quick-Start Option, or have not agreed to the Quick-Start rate request. TCP host B communicates the final rate request to TCP host A in a transport-level Quick-Start Response in an answering TCP packet.

If the Quick-Start Request is approved by all routers along the path, then the TCP host can send at up to the approved rate for a window of data. Subsequent transmissions will be governed by the default TCP congestion control mechanisms of that connection. If the Quick-Start Request is not approved, then the sender would use the default congestion control mechanisms.

Quick-Start would not be the first mechanism for explicit communication from routers to transport protocols about sending rates. Explicit Congestion Notification (ECN) gives explicit congestion control feedback from routers to transport protocols, based on the router detecting congestion before buffer overflow [RFC3168]. In contrast, routers would not use Quick-Start to give congestion information, but instead would use Quick-Start as an optional mechanism to give permission to transport protocols to use higher sending rates, based on the ability of all the routers along the path to determine if their respective output links are significantly underutilized.

Section 2 gives an overview of Quick-Start. The formal specifications for Quick-Start are contained in Sections 3, 4, 6.1.1, and 6.3. In particular, Quick-Start is specified for IPv4 and for IPv6 in Section 3, and is specified for TCP in Section 4. Section 6 consists mostly of a non-normative discussion of interactions of Quick-Start with IP tunnels and MPLS; however, Section 6.1.1 and 6.3 specify behavior for IP tunnels that are aware of Quick-Start.

The rest of the document is non-normative, as follows. Section 5 shows that Quick-Start is compatible with IPsec AH (Authentication Header). Section 7 gives a non-normative set of guidelines for specifying Quick-Start in other transport protocols, and Section 8 discusses using Quick-Start in transport end-nodes and routers. Section 9 gives an evaluation of the costs and benefits of Quick-Start, and Section 10 discusses implementation and deployment issues. The appendices discuss related work, Quick-Start design decisions, and possible router algorithms.

1.1. Conventions and Terminology

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].

2. Assumptions and General Principles

This section describes the assumptions and general principles behind the design of the Quick-Start mechanism.

Assumptions:

* The data transfer in the two directions of a connection traverses different queues, and possibly even different routers. Thus, any mechanism for determining the allowed sending rate would have to be used independently for each direction.

* The path between the two endpoints is relatively stable, such that the path used by the Quick-Start Request is generally the same path used by the Quick-Start packets one round-trip time later. [ZDPS01] shows this assumption should be generally valid. However, [RFC3819] discusses a range of Bandwidth on Demand subnets that could cause the characteristics of the path to change over time.

* Any new mechanism must be incrementally deployable and might not be supported by all the routers and/or end-hosts. Thus, any new mechanism must be able to accommodate non-supporting routers or end-hosts without disturbing the current Internet semantics. We note that, while Quick-Start is incrementally deployable in this sense, a Quick-Start Request cannot be approved for a particular connection unless both end-nodes and all the routers along the path have been configured to support Quick-Start.

General Principles:

* Our underlying premise is that explicit feedback from all the routers along the path would be required, in the current architecture, for best-effort connections to use initial windows significantly larger than those allowed by [RFC3390], in the absence of other information about the path.

* A router should only approve a Quick-Start Request if the output link is underutilized. Any other approach will result in either per-flow state at the router, or the possibility of a (possibly transient) queue at the router.

* No per-flow state should be required at the router. Note that, while per-flow state is not required, we also do not preclude a router from storing per-flow state for making Quick-Start decisions or for checking for misbehaving nodes.

2.1. Overview of Quick-Start

In this section, we give an overview of the use of Quick-Start with TCP to request a higher congestion window. The description in this section is non-normative; the normative description of Quick-Start with IP and TCP follows in Sections 3 and 4. Quick-Start could be used in the middle of a connection, e.g., after an idle or underutilized period, as well as for the initial sending rate; these uses of Quick-Start are discussed later in the document.

Quick-Start requires end-points and routers to work together, with end-points requesting a higher sending rate in the Quick-Start (QS) option in IP, and routers along the path approving, modifying, discarding, or ignoring (and therefore disallowing) the Quick-Start Request. The receiver uses reliable, transport-level mechanisms to inform the sender of the status of the Quick-Start Request. For example, when TCP is used, the TCP receiver sends feedback to the sender using a Quick-Start Response option in the TCP header. In addition, Quick-Start assumes a unicast, congestion-controlled transport protocol; we do not consider the use of Quick-Start for multicast traffic.

When sent as a request, the Quick-Start Option includes a request for a sending rate in bits per second, and a Quick-Start Time to Live (QS TTL) to be decremented by every router along the path that understands the option and approves the request. The Quick-Start TTL is initialized by the sender to a random value. The transport receiver returns the rate, information about the TTL, and the Quick-Start Nonce to the sender using transport-level mechanisms; for TCP, the receiver sends this information in the Quick-Start Response in the TCP header. In particular, the receiver computes the difference between the Quick-Start TTL and the IP TTL (the TTL in the IP header) of the Quick-Start Request packet, and returns this in the Quick-Start Response. The sender uses the TTL difference to determine if all the routers along the path decremented the Quick-Start TTL, approving the Quick-Start Request.

If the request is approved by all the routers along the path, then the TCP sender combines this allowed rate with the measurement of the round-trip time, and ends up with an allowed TCP congestion window. This window is sent rate-paced over the next round-trip time, or until an ACK packet is received.

Figure 1 shows a successful use of Quick-Start, with the sender's IP layer and both routers along the path approving the Quick-Start Request, and the TCP receiver using the Quick-Start Response to return information to the TCP sender. In this example, Quick-Start is used by TCP to establish the initial congestion window.

```
   Sender        Router 1       Router 2          Receiver
   ------        --------       --------          --------
     | <IP TTL: 63>
     | <QS TTL: 91>
     | <TTL Diff: 28>
     | Quick-Start Request
     | in SYN or SYN/ACK.
     | IP: Decrement QS TTL
     | to approve request -->
     |
     |              Decrement
     |              QS TTL
     |              to approve
     |              request -->
     |
     |                             Decrement
     |                             QS TTL
     |                             to approve
     |                             request -->
     |
     |                                          <IP TTL: 60>
     |                                          <QS TTL: 88>
     |                                          <TTL Diff: 28>
     |                                          Return Quick-Start
     |                                           info to sender in
     |                                          Quick-Start Response
     |                                         <-- in TCP ACK packet.
     |
     | <TTL Diff: 28>
     | Quick-Start approved,
     | translate to cwnd.
     | Report Approved Rate.
     V Send cwnd paced over one RTT. -->
```

Figure 1: A Successful Quick-Start Request.

Figure 2 shows an unsuccessful use of Quick-Start, with one of the routers along the path not approving the Quick-Start Request. If the Quick-Start Request is not approved, then the sender uses the default congestion control mechanisms for that transport protocol, including the default initial congestion window, response to idle periods, etc.

```
   Sender        Router 1       Router 2          Receiver
   ------        --------       --------          --------
   | <IP TTL: 63>
   | <QS TTL: 91>
   | <TTL Diff: 28>
   | Quick-Start Request
   | in SYN or SYN/ACK.
   | IP: Decrement QS TTL
   | to approve request -->
   |
   |              Decrement
   |              QS TTL
   |              to approve
   |              request -->
   |
   |                             Forward packet
   |                             without modifying
   |                             Quick-Start Option. -->
   |
   |                                         <IP TTL: 60>
   |                                         <QS TTL: 89>
   |                                         <TTL Diff: 29>
   |                                         Return Quick-Start
   |                                          info to sender in
   |                                         Quick-Start Response
   |                                       <-- in TCP ACK packet.
   |
   | <TTL Diff: 29>
   | Quick-Start not approved.
   | Report approved rate.
   V Use default initial cwnd. -->
```

Figure 2: An Unsuccessful Quick-Start Request.

3. The Quick-Start Option in IP

3.1. The Quick-Start Option for IPv4

The Quick-Start Request for IPv4 is defined as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Option      |  Length=8     | Func. | Rate  |   QS TTL      |
|               |               | 0000  |Request|               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        QS Nonce                           | R |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: The Quick-Start Option for IPv4.
A Quick-Start Request.

The first byte contains the option field, which includes the one-bit copy flag, the 2-bit class field, and the 5-bit option number.

The second byte contains the length field, indicating an option length of eight bytes.

The third byte includes a four-bit Function field. If the Function field is set to "0000", then the Quick-Start Option is a Rate Request. In this case, the second half of the third byte is a four-bit Rate Request field.

For a Rate Request, the fourth byte contains the Quick-Start TTL (QS TTL) field. The sender MUST set the QS TTL field to a random value. Routers that approve the Quick-Start Request decrement the QS TTL (mod 256) by the same amount that they decrement the IP TTL. (As elsewhere in this document, we use the term 'router' imprecisely to also include the Quick-Start functionality at the IP layer at the sender.) The QS TTL is used by the sender to detect if all the routers along the path understood and approved the Quick-Start option.

For a Rate Request, the transport sender MUST calculate and store the TTL Diff, the difference between the IP TTL value, and the QS TTL value in the Quick-Start Request packet, as follows:

TTL Diff = (IP TTL - QS TTL) mod 256 (1)

For a Rate Request, bytes 5-8 contain a 30-bit QS Nonce, discussed in Section 3.4, and a two-bit Reserved field. The sender SHOULD set the reserved field to zero, and routers and receivers SHOULD ignore the reserved field. The sender SHOULD set the 30-bit QS Nonce to a random value.

The sender initializes the Rate Request to the desired sending rate, including an estimate of the transport and IP header overhead. The encoding function for the Rate Request sets the request rate to K*2^N bps (bits per second), for N the value in the Rate Request field, and for K set to 40,000. For N=0, the rate request would be set to zero, regardless of the encoding function. This is illustrated in Table 1 below. For the four-bit Rate Request field, the request range is from 80 Kbps to 1.3 Gbps. Alternate encodings that were considered for the Rate Request are given in Appendix B.2.

N	Rate Request (in Kbps)
0	0
1	80
2	160
3	320
4	640
5	1,280
6	2,560
7	5,120
8	10,240
9	20,480
10	40,960
11	81,920
12	163,840
13	327,680
14	655,360
15	1,310,720

Table 1: Mapping from Rate Request Field to Rate Request in Kbps.

Routers can approve the Quick-Start Request for a lower rate by decreasing the Rate Request in the Quick-Start Request. Section 4.2 discusses the Quick-Start Response from the TCP receiver to the TCP sender, and Section 4.4 discusses the TCP sender's mechanism for determining if a Quick-Start Request has been approved.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Option      |  Length=8     | Func. | Rate  |   Not Used    |
|               |               | 1000  | Report|               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        QS Nonce                           | R |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: The Quick-Start Option for IPv4.
Report of Approved Rate.

If the Function field in the third byte of the Quick-Start Option is set to "1000", then the Quick-Start Option is a Report of Approved Rate. In this case, the second four bits in the third byte are the Rate Report field, formatted exactly as in the Rate Request field in a Rate Request. For a Report of Approved Rate, the fourth byte of the Quick-Start Option is not used. Bytes 5-8 contain a 30-bit QS Nonce and a 2-bit Reserved field.

After an approved Rate Request, the sender MUST report the Approved Rate, using a Quick-Start Option configured as a Report of Approved Rate with the Rate Report field set to the approved rate, and the QS Nonce set to the QS Nonce sent in the Quick-Start Request. The packet containing the Report of Approved Rate MUST be either a control packet sent before the first Quick-Start data packet, or a Quick-Start Option in the first data packet itself. The Report of Approved Rate does not have to be sent reliably; for example, if the approved rate is reported in a separate control packet, the sender does not necessarily know if the control packet has been dropped in the network. If the packet containing the Quick-Start Request is acknowledged, but the acknowledgement packet does not contain a Quick-Start Response, then the sender MUST assume that the Quick-Start Request was denied, and set a Report of Approved Rate with a rate of zero. Routers may choose to ignore the Report of Approved Rate, or to use the Report of Approved Rate but ignore the QS Nonce. Alternately, some routers that use the Report of Approved Rate may choose to match the QS Nonce, masked by the approved rate, with the QS Nonce seen in the original request.

If the Rate Request is denied, the sender MUST send a Report of Approved Rate with the Rate Report field set to zero.

We note that, unlike a Quick-Start Request sent at the beginning of a connection, when a Quick-Start Request is sent in the middle of a connection, the connection could already have an established congestion window or sending rate. The Rate Request is the requested total rate for the connection, including the current rate of the connection; the Rate Request is *not* a request for an additional sending rate over and above the current sending rate. If the Rate Request is denied, or lowered to a value below the connection's current sending rate, then the sender ignores the request, and reverts to the default congestion control mechanisms of the transport protocol.

The use of the Quick-Start Option does not require the additional use of the Router Alert Option [RFC2113].

We note that in IPv4, a change in IP options at routers requires recalculating the IP header checksum.

3.2. The Quick-Start Option for IPv6

The Quick-Start Option for IPv6 is placed in the Hop-by-Hop Options extension header that is processed at every network node along the communication path [RFC2460]. The option format following the generic Hop-by-Hop Options header is identical to the IPv4 format, with the exception that the Length field should exclude the common type and length fields in the option format and be set to 6 bytes instead of 8 bytes.

For a Quick-Start Request, the transport receiver compares the Quick-Start TTL with the IPv6 Hop Limit field to calculate the TTL Diff. (The Hop Limit in IPv6 is the equivalent of the TTL in IPv4.) That is, TTL Diff MUST be calculated and stored as follows:

TTL Diff = (IPv6 Hop Limit - QS TTL) mod 256 (2)

Unlike IPv4, modifying or deleting the Quick-Start IPv6 Option does not require checksum re-calculation, because the IPv6 header does not have a checksum field, and modifying the Quick-Start Request in the IPv6 Hop-by-Hop options header does not affect the IPv6 pseudo-header checksum used in upper-layer checksum calculations.

Appendix A of RFC 2460 requires that all packets with the same flow label must be originated with the same hop-by-hop header contents, which would be incompatible with Quick-Start. However, a later IPv6 flow label specification [RFC3697] updates the use of flow labels in IPv6 and removes this restriction. Therefore, Quick-Start is compatible with the current IPv6 specifications.

3.3. Processing the Quick-Start Request at Routers

The Quick-Start Request does not report the current sending rate of the connection sending the request; in the default case of a router (or IP-layer implementation at an end-node) that does not maintain per-flow state, a router makes the conservative assumption that the flow's current sending rate is zero. Each participating router can either terminate or approve the Quick-Start Request. A router MUST only approve a Quick-Start Request if the output link is underutilized, and if the router judges that the output link will continue to be underutilized if this and earlier approved requests are used by the senders. Otherwise, the router reduces or terminates the Quick-Start Request.

While the paragraph above defines the *semantics* of approving a Quick-Start Request, this document does not specify the specific algorithms to be used by routers in processing Quick-Start Requests or Reports. This is similar to RFC 3168, which specifics the semantics of the ECN codepoints in the IP header, but does not specify specific algorithms for routers to use in deciding when to drop or mark packets before buffer overflow.

A router that wishes to terminate the Quick-Start Request SHOULD either delete the Quick-Start Request from the IP header or zero the QS TTL, QS Nonce, and Rate Request fields. Deleting the Quick-Start Request saves resources because downstream routers will have no option to process, but zeroing the Rate Request field may be more efficient for routers to implement. As suggested in [B05], future additions to Quick-Start could define error codes for routers to insert into the QS Nonce field to report back to the sender the reason that the Quick-Start Request was denied, e.g., that the router is denying all Quick-Start Requests at this time, or that this router, as a matter of policy, does not grant Quick-Start requests. A router that doesn't understand the Quick-Start Option will simply forward the packet with the Quick-Start Request unchanged (ensuring that the TTL Diff will not match and Quick-Start will not be used).

If the participating router has decided to approve the Quick-Start Request, it does the following:

* The router MUST decrement the QS TTL by the amount the IP TTL is decremented (usually one).

* If the router is only willing to approve a Rate Request less than that in the Quick-Start Request, then the router replaces the Rate Request with a smaller value. The router MUST NOT increase the Rate Request in the Quick-Start Request. If the router decreases

the Rate Request, the router MUST also modify the QS Nonce, as described in Section 3.4.

* In IPv4, the router MUST update the IP header checksum.

If the router approves the Quick-Start Request, this approval SHOULD be taken into account in the router's decision to accept or reject subsequent Quick-Start Requests (e.g., using a variable that tracks the recent aggregate of accepted Quick-Start Requests). This consideration of earlier approved Quick-Start Requests is necessary to ensure that the router only approves a Quick-Start Request when the router judges that the output link will remain underutilized if this and earlier Quick-Start Requests are used by the senders.

In addition, the approval of a Quick-Start Request SHOULD NOT be used by the router to affect the treatment of the data packets that arrive from this connection in the next few round-trip times. That is, the approval by the router of a Quick-Start Request does not give differential treatment for Quick-Start data packets at that router; it is only a statement from the router that the router believes that the subsequent Quick-Start data packets from this connection will not change the current underutilized state of the router.

A non-participating router forwards the Quick-Start Request unchanged, without decrementing the QS TTL. The non-participating router still decrements the TTL field in the IP header, as is required for all routers [RFC1812]. As a result, the sender will be able to detect that the Quick-Start Request had not been understood or approved by all of the routers along the path.

A router that uses multipath routing for packets within a single connection MUST NOT approve a Quick-Start Request. This is discussed in more detail in Section 9.2.

3.3.1. Processing the Report of Approved Rate

If the Quick-Start Option has the Function field set to "1000", then this is a Report of Approved Rate, rather than a Rate Request. The router MAY ignore such an option, and, in any case, it MUST NOT modify the contents of the option for a Report of Approved Rate. However, the router MAY use the Approved Rate report to check that the sender is not lying about the approved rate. If the reported Approved Rate is higher than the rate that the router actually approved for this connection in the previous round-trip time, then the router may implement some policy for cheaters. For instance, the router MAY decide to deny future Quick-Start Requests from this sender, including, if desired, deleting Quick-Start Requests from future packets from this sender. Section 9.4.1 discusses misbehaving senders in more detail. From the Report of Approved Rate, the router can also learn if some of the downstream routers have approved the Quick-Start Request for a smaller rate or denied the use of Quick-Start, and adjust its bandwidth allocations accordingly.

3.4. The QS Nonce

The QS Nonce gives the Quick-Start sender some protection against receivers lying about the value of the received Rate Request. This is particularly important if the receiver knows the original value of the Rate Request (e.g., when the sender always requests the same value, and the receiver has a long history of communication with that sender). Without the QS Nonce, there is nothing to prevent the receiver from reporting back to the sender a Rate Request of K, when the received Rate Request was, in fact, less than K.

Table 2 gives the format for the 30-bit QS Nonce.

Bits	Purpose
Bits 0-1:	Rate 15 -> Rate 14
Bits 2-3:	Rate 14 -> Rate 13
Bits 4-5:	Rate 13 -> Rate 12
Bits 6-7:	Rate 12 -> Rate 11
Bits 8-9:	Rate 11 -> Rate 10
Bits 10-11:	Rate 10 -> Rate 9
Bits 12-13:	Rate 9 -> Rate 8
Bits 14-15:	Rate 8 -> Rate 7
Bits 16-17:	Rate 7 -> Rate 6
Bits 18-19:	Rate 6 -> Rate 5
Bits 20-21:	Rate 5 -> Rate 4
Bits 22-23:	Rate 4 -> Rate 3
Bits 24-25:	Rate 3 -> Rate 2
Bits 26-27:	Rate 2 -> Rate 1
Bits 28-29:	Rate 1 -> Rate 0

Table 2: The QS Nonce.

The transport sender MUST initialize the QS Nonce to a random value. If the router reduces the Rate Request from rate K to rate K-1, then the router MUST set the field in the QS Nonce for "Rate K -> Rate K-1" to a new random value. Similarly, if the router reduces the Rate Request by N steps, the router MUST set the 2N bits in the relevant fields in the QS Nonce to a new random value. The receiver MUST report the QS Nonce back to the sender.

If the Rate Request was not decremented in the network, then the QS Nonce should have its original value. Similarly, if the Rate Request was decremented by N steps in the network, and the receiver reports back a Rate Request of K, then the last 2K bits of the QS Nonce should have their original value.

With the QS Nonce, the receiver has a 1/4 chance of cheating about each step change in the rate request. Thus, if the rate request is reduced by two steps in the network, the receiver has a 1/16 chance of successfully reporting that the original request was approved, as this requires reporting the original value for the QS nonce. Similarly, if the rate request is reduced many steps in the network, and the receiver receives a QS Option with a rate request of K, the receiver has a 1/16 chance of guessing the original values for the fields in the QS nonce for "Rate K+2 -> Rate K+1" and "Rate K+1 -> Rate K". Thus, the receiver has a 1/16 chance of successfully lying and saying that the received rate request was K+2 instead of K.

We note that the protection offered by the QS Nonce is the same whether one router makes all the decrements in the rate request, or whether they are made at different routers along the path.

The requirements for randomization for the sender and routers in setting 'random' values in the QS Nonce are not stringent -- almost any form of pseudo-random numbers will do. The requirement is that the original value for the QS Nonce is not easily predictable by the receiver, and in particular, the nonce MUST NOT be easily determined from inspection of the rest of the packet or from previous packets. In particular, the nonce MUST NOT be based only on a combination of specific packet header fields. Thus, if two bits of the QS Nonce are changed by a router along the path, the receiver should not be able to guess those two bits from the other 28 bits in the QS Nonce.

An additional requirement is that the receiver cannot be able to tell, from the QS Nonce itself, which numbers in the QS Nonce were generated by the sender, and which were generated by routers along the path. This makes it harder for the receiver to infer the value of the original rate request, making it one step harder for the receiver to cheat.

Section 9.4 also considers issues of receiver cheating in more detail.

4. The Quick-Start Mechanisms in TCP

This section describes how the Quick-Start mechanism would be used in TCP. We first sketch the procedure and then tightly define it in the subsequent subsections.

If a TCP sender (say, host A) would like to use Quick-Start, the TCP sender puts the requested sending rate in bits per second, appropriately formatted, in the Quick-Start Option in the IP header of the TCP packet, called the Quick-Start Request packet. (We will be somewhat loose in our use of "packet" vs. "segment" in this section.) When used for initial start-up, the Quick-Start Request packet can be either the SYN or SYN/ACK packet, as illustrated in Figure 1. The requested rate includes an estimate for the transport and IP header overhead. The TCP receiver (say, host B) returns the Quick-Start Response option in the TCP header in the responding SYN/ACK packet or ACK packet, called the Quick-Start Response packet, informing host A of the results of their request.

If the acknowledging packet does not contain a Quick-Start Response, or contains a Quick-Start Response with the wrong value for the TTL Diff or the QS Nonce, then host A MUST assume that its Quick-Start request failed. In this case, host A sends a Report of Approved Rate with a Rate Report of zero, and uses TCP's default congestion control procedure. For initial start-up, host A uses the default initial congestion window ([RFC2581], [RFC3390]).

If the returning packet contains a valid Quick-Start Response, then host A uses the information in the response, along with its measurement of the round-trip time, to determine the Quick-Start congestion window (QS-cwnd). Quick-Start data packets are defined as data packets sent as the result of a successful Quick-Start request, up to the time when the first Quick-Start packet is acknowledged. The sender also sends a Report of Approved Rate. In order to use Quick-Start, the TCP host MUST use rate-based pacing [VH97] to transmit Quick-Start packets at the rate indicated in the Quick-Start Response, at the level of granularity possible by the sending host. We note that the limitations of interrupt timing on computers can limit the ability of the TCP host in rate-pacing the outgoing packets.

The two TCP end-hosts can independently decide whether to request Quick-Start. For example, host A could send a Quick-Start Request in the SYN packet, and host B could also send a Quick-Start Request in the SYN/ACK packet.

4.1. Sending the Quick-Start Request

When sending a Quick-Start Request, the TCP sender SHOULD send the request on a packet that requires an acknowledgement, such as a SYN, SYN/ACK, or data packet. In this case, if the packet is acknowledged but no Quick-Start Response is included, then the sender knows that the Quick-Start Request has been denied, and can send a Report of Approved Rate.

In addition to the use of Quick-Start when a connection is established, there are several additional points in a connection when a transport protocol may want to issue a Rate Request. We first reiterate the notion that Quick-Start is a coarse-grained mechanism. That is, Quick-Start's Rate Requests are not meant to be used for fine-grained control of the transport's sending rate. Rather, the transport MAY issue a Rate Request when no information about the appropriate sending rate is available, and the default congestion control mechanisms might be significantly underestimating the appropriate sending rate.

The following are potential points where Quick-Start may be useful:

(1) At or soon after connection initiation, when the transport has no idea of the capacity of the network path, as discussed above. (A transport that uses TCP Control Block sharing [RFC2140], the Congestion Manager [RFC3124], or other mechanisms for sharing congestion information may not need Quick-Start to determine an appropriate rate.)

(2) After an idle period when the transport no longer has a validated estimate of the available bandwidth for this flow. (An example could be a persistent-HTTP connection when a new HTTP request is received after an idle period.)

(3) After a host has received explicit indications that one of the endpoints has moved its point of network attachment. This can happen due to some underlying mobility mechanism like Mobile IP ([RFC3344], [RFC3775]). Some transports, such as Steam Control Transmission Protocol (SCTP) [RFC2960], may associate with multiple IP addresses and can switch addresses (and therefore network paths) in mid-connection. If the transport has concrete knowledge of a changing network path, then the current sending rate may not be appropriate, and the transport sender may use Quick-Start to probe the network to see if it can send at a higher rate. (Alternatively, traditional slow-start should be used in this case when Quick-Start is not available.)

(4) After an application-limited period, when the sender has been using only a small amount of its appropriate share of the network capacity and has no valid estimate for its fair share. In this case, Quick-Start may be an appropriate mechanism to determine if the sender can send at a higher rate. For instance, consider an application that steadily exchanges low- rate control messages and suddenly needs to transmit a large amount of data.

Of the above, this document recommends that a TCP sender MAY attempt to use Quick-Start in cases (1) and (2). It is NOT RECOMMENDED that a TCP sender use Quick-Start for case (3) at the current time. Case (3) requires external notifications not presently defined for TCP or other transport protocols. Finally, a TCP SHOULD NOT use Quick-Start for case (4) at the current time. Case (4) requires further thought and investigation with regard to how the transport protocol could determine it was in a situation that would warrant transmitting a Quick-Start Request.

As a general guideline, a TCP sender SHOULD NOT request a sending rate larger than it is able to use over the next round-trip time of the connection (or over 100 ms, if the round-trip time is not known), except as required to round up the desired sending rate to the next-highest allowable request.

In any circumstances, the sender MUST NOT make a QS request if it has made a QS request within the most recent round-trip time.

Section 4.7 discusses some of the issues of using Quick-Start at connection initiation, and Section 4.8 discusses issues that arise when Quick-Start is used to request a larger sending rate after an idle period.

4.2. The Quick-Start Response Option in the TCP header

In order to approve the use of Quick-Start, the TCP receiver responds to the receipt of a Quick-Start Request with a Quick-Start Response, using the Quick-Start Response Option in the TCP header. TCP's Quick-Start Response option is defined as follows:

```
     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      |  Length=8     | Resv. | Rate  |   TTL Diff    |
    |               |               |       |Request|               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                   QS Nonce                                | R |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: The Quick-Start Response Option in the TCP Header.

The first byte of the Quick-Start Response option contains the option kind, identifying the TCP option.

The second byte of the Quick-Start Response option contains the option length in bytes. The length field MUST be set to 8 bytes.

The third byte of the Quick-Start Response option contains a four-bit Reserved field, and the four-bit allowed Rate Request, formatted as in the Quick-Start Rate Request option.

The fourth byte of the TCP option contains the TTL Diff. The TTL Diff contains the difference between the IP TTL and QS TTL fields in the received Quick-Start Request packet, as calculated in equations (1) or (2) (depending on whether IPv4 or IPv6 is used).

Bytes 5-8 of the TCP option contain the 30-bit QS Nonce and a two-bit Reserved field.

We note that, while there are limitations on the potential size of the Quick-Start Response Option, a Quick-Start Response Option of eight bytes should not be a problem. The TCP Options field can contain up to 40 bytes. Other TCP options that might be used in a SYN or SYN/ACK packet include Maximum Segment Size (four bytes), Time Stamp (ten bytes), Window Scale (three bytes), and Selective Acknowledgments Permitted (two bytes).

4.3. TCP: Sending the Quick-Start Response

An end host (say, host B) that receives an IP packet containing a Quick-Start Request passes the Quick-Start Request, along with the value in the IP TTL field, to the receiving TCP layer.

If the TCP host is willing to permit the Quick-Start Request, then a Quick-Start Response option is included in the TCP header of the corresponding acknowledgement packet. The Rate Request in the Quick-Start Response option is set to the received value of the Rate Request in the Quick-Start Option, or to a lower value if the TCP receiver is only willing to allow a lower Rate Request. The TTL Diff in the Quick-Start Response is set to the difference between the IP TTL value and the QS TTL value as given in equation (1) or (2) (depending on whether IPv4 or IPv6 is used). The QS Nonce in the Response is set to the received value of the QS Nonce in the Quick-Start Option.

If an end host receives an IP packet with a Quick-Start Request with a rate request of zero, then that host SHOULD NOT send a Quick-Start Response.

The Quick-Start Response MUST NOT be resent if it is lost in the network. Packet loss could be an indication of congestion on the return path, in which case it is better not to approve the Quick-Start Request.

4.4. TCP: Receiving and Using the Quick-Start Response Packet

A TCP host (say, TCP host A) that sent a Quick-Start Request and receives a Quick-Start Response in an acknowledgement first checks that the Quick-Start Response is valid. The Quick-Start Response is valid if it contains the correct value for the TTL Diff, and an equal or lesser value for the Rate Request than that transmitted in the Quick-Start Request. In addition, if the received Rate Request is K, then the rightmost 2K bits of the QS Nonce must match those bits in the QS Nonce sent in the Quick-Start Request. If these checks are not successful, then the Quick-Start Request failed, and the TCP host MUST use the default TCP congestion window that it would have used without Quick-Start. If the rightmost 2K bits of the QS Nonce do not match those bits in the QS Nonce sent in the Quick-Start Request, for a received Rate Request of K, then the TCP host MUST NOT send additional Quick-Start Requests during the life of the connection. Whether or not the Quick-Start Request was successful, the host receiving the Quick-Start Response MUST send a Report of Approved Rate. Similarly, if the packet containing the Quick-Start Request is acknowledged, but the acknowledgement does not include a Quick-Start Response, then the sender MUST send a Report of Approved Rate.

If the checks of the TTL Diff and the Rate Request are successful, and the TCP host is going to use the Quick-Start Request, it MUST start using it within one round-trip time of receiving the Quick-Start Response, or within three seconds, whichever is smaller. To use the Quick-Start Request, the host sets its Quick-Start congestion window (in terms of MSS-sized segments), QS-cwnd, as follows:

```
QS-cwnd = (R * T) / (MSS + H)                               (3)
```

where R is the Rate Request in bytes per second, T is the measured round-trip time in seconds, and H is the estimated TCP/IP header size in bytes (e.g., 40 bytes).

Derivation: the sender is allowed to transmit at R bytes per second including packet headers, but only R*MSS/(MSS+H) bytes per second, or equivalently R*T*MSS/(MSS+H) bytes per round-trip time, of application data.

The TCP host SHOULD set its congestion window cwnd to QS-cwnd only if QS-cwnd is greater than cwnd; otherwise, QS-cwnd is ignored. If QS-cwnd is used, the TCP host sets a flag that it is in Quick-Start mode, and while in Quick-Start mode, the TCP sender MUST use rate-based pacing to pace out Quick-Start packets at the approved rate. If, during Quick-Start mode, the TCP sender receives ACKs for packets sent before this Quick-Start mode was entered, these ACKs are processed as usual, following the default congestion control mechanisms. Quick-Start mode ends when the TCP host receives an ACK for one of the Quick-Start packets.

If the congestion window has not been fully used when the first ack arrives ending the Quick-Start mode, then the congestion window is decreased to the amount that has actually been used so far. This is necessary because when the Quick-Start Response is received, the TCP sender's round-trip-time estimate might be longer than for succeeding round-trip times, e.g., because of delays at routers processing the IP Quick-Start Option, or because of delays at the receiver in responding to the Quick-Start Request packet. In this case, an overly large round-trip-time estimate could have caused the TCP sender to translate the approved Quick-Start sending rate in bytes per second into a congestion window that is larger than needed, with the TCP sender receiving an ACK for the first Quick- Start packet before the entire congestion window has been used. Thus, when the TCP sender receives the first ACK for a Quick-Start packet, the sender MUST reduce the congestion window to the amount that has actually been used.

As an example, a TCP sender with an approved Quick-Start Request of R KBps, B-byte packets including headers, and an RTT estimate of T seconds, would translate the Rate Request of R KBps to a congestion window of R*T/B packets. The TCP sender would send the Quick-Start packets rate-paced at R KBps. However, if the actual current round-trip time was T/2 seconds instead of T seconds, then the sender would begin to receive acknowledgements for Quick-Start packets after T/2 seconds. Following the paragraph above, the TCP sender would then reduce its congestion window from R*T/B to approximately R*T/(B*2) packets, the actual number of packets that were needed to fill the pipe at a sending rate of R KBps. (Note: this is just an illustration; the congestion window is actually set to the amount of data sent before the ACK arrives and not based on equations above.)

After Quick-Start mode is exited and the congestion window adjusted if necessary, the TCP sender returns to using the default congestion-control mechanisms, processing further incoming ACK packets as specified by those congestion control mechanisms. For example, if the TCP sender was in slow-start prior to the Quick-Start Request, and no packets were lost or marked since that time, then the sender continues in slow-start after exiting Quick-Start mode, as allowed by ssthresh.

To add robustness, the TCP sender MUST use Limited Slow-Start [RFC3742] along with Quick-Start. With Limited Slow-Start, the TCP sender limits the number of packets by which the congestion window is increased for one window of data during slow-start.

When Quick-Start is used at the beginning of a connection, before any packet marks or losses have been reported, the TCP host MAY use the reported Rate Request to set the slow-start threshold to a desired value, e.g., to some small multiple of the congestion window. A possible future research topic is how the sender might modify the slow-start threshold at the beginning of a connection to avoid overshooting the path capacity. (The initial value of ssthresh is allowed to be arbitrarily high, and some TCP implementations use the size of the advertised window for ssthresh [RFC2581].)

4.5. TCP: Controlling Acknowledgement Traffic on the Reverse Path

When a Quick-Start Request is approved for a TCP sender, the resulting Quick-Start data traffic can result in a sudden increase in traffic for pure ACK packets on the reverse path. For example, for the largest Quick-Start Request of 1.3 Gbps, given a TCP sender with 1500-byte packets and a TCP receiver with delayed acknowledgements acking every other packet, this could result in 17.3 Mbps of acknowledgement traffic on the reverse path.

One possibility, in cases with large Quick-Start Requests, would be for TCP receivers to send Quick-Start Requests to request bandwidth for the acknowledgement traffic on the reverse path. However, in our view, a better approach would be for TCP receivers to simply control the rate of sending acknowledgement traffic. The optimal future solution would involve the explicit use of congestion control for TCP acknowledgement traffic, as is done now for the acknowledgement traffic in DCCP's CCID2 [RFC4341].

In the absence of congestion control for acknowledgement traffic, the TCP receiver could limit its sending rate for ACK packets sent in response to Quick-Start data packets. The following information is needed by the TCP receiver:

* The RTT: TCP naturally measures the RTT of the path and therefore should have a sample of the RTT. If the TCP receiver does not have a measurement of the round-trip time, it can use the time between the receipt of the Quick-Start Request and the Report of Approved Rate.

* The Approved Rate Request (R): When the TCP receiver receives the Quick-Start Response packet, the receiver knows the value of the approved Rate Request.

* The MSS: TCP advertises the MSS during the initial three-way handshake; therefore, the receiver should have an understanding of the packet size the sender will be using. If the receiver does not have such an understanding or wishes to confirm the negotiated MSS, the size of the first data packet can be used.

With this set of information, the TCP receiver can restrict its sending rate for pure acknowledgment traffic to at most 100 pure ACK packets per RTT by sending at most one ACK for every K data packets, for the ACK Ratio K set to R*RTT/(100*8*MSS). The receiver would acknowledge the first Quick-Start data packet, and every succeeding K data packets. Thus, for a somewhat extreme case of R=1.3 Gbps, RTT=0.2 seconds, and MSS=1500 bytes, K would be set to 216, and the receiver would acknowledge every 216 data packets. From [RFC2581], the ACK Ratio K should have a minimum value of two. When the ACK Ratio is greater than two, and the TCP sender receives acknowledgements each acknowledging more than two data packets, the TCP sender may want to use rate-based pacing to control the burstiness of its outgoing data traffic.

In the absence of explicit congestion control mechanisms, the TCP end nodes cannot determine the packet drop rate for pure acknowledgement traffic. This is true with or without Quick-Start. However, the TCP receiver could limit its increase in the sending rate for pure ACK packets by at most doubling the sending rate for pure ACK packets from one round-trip time to the next. The TCP receiver would do this by halving the ACK Ratio each round-trip time.

Note that the above is one particular mechanism that could be used to control the ACK stream. Future work that investigates this scheme and others in detail is encouraged.

4.6. TCP: Responding to a Loss of a Quick-Start Packet

For TCP, we have defined a "Quick-Start packet" as one of the packets sent in the window immediately following a successful Quick-Start Request. After detecting the loss or ECN-marking of a Quick-Start packet, TCP MUST revert to the default congestion control procedures that would have been used if the Quick-Start Request had not been approved. For example, if Quick-Start is used for setting the initial window, and a packet from the initial window is lost or marked, then the TCP sender MUST then slow-start with the default initial window that would have been used if Quick-Start had not been used. In addition to reverting to the default congestion control mechanisms, the sender MUST take into account that the Quick-Start congestion window was too large. Thus, the sender SHOULD decrease ssthresh to, at most, half the number of Quick-Start packets that were successfully transmitted. Appendix B.5 discusses possible alternatives in responding to the loss of a Quick-Start packet.

If a Quick-Start packet is lost or ECN-marked, then the sender SHOULD NOT make future Quick-Start Requests for this connection.

We note that ECN [RFC3168] MAY be used with Quick-Start. As is always the case with ECN, the sender's congestion control response to an ECN-marked Quick-Start packet is the same as the response to a dropped Quick-Start packet, thus reverting to slow start in the case of Quick-Start packets marked as experiencing congestion.

4.7. TCP: A Quick-Start Request for a Larger Initial Window

Some of the issues of using Quick-Start are related to the specific scenario in which Quick-Start is used. This section discusses the following issues that arise when Quick-Start is used by TCP to request a larger initial window: (1) interactions with Path MTU Discovery (PMTUD); and (2) Quick-Start Request packets that are discarded by middleboxes.

4.7.1. Interactions with Path MTU Discovery

One issue when Quick-Start is used to request a large initial window concerns the interactions between the large initial window and Path MTU Discovery. Some of the issues are discussed in RFC 3390:

"When larger initial windows are implemented along with Path MTU Discovery [RFC1191], alternatives are to set the 'Don't Fragment' (DF) bit in all segments in the initial window, or to set the 'Don't Fragment' (DF) bit in one of the segments. It is an open question as to which of these two alternatives is best."

If the sender knows the Path MTU when the initial window is sent (e.g., from a PMTUD cache or from some other IETF-approved method), then the sender SHOULD use that MTU for segments in the initial window. Unfortunately, the sender doesn't necessarily know the Path MTU when it sends packets in the initial window. In this case, the sender should be conservative in the packet size used. Sending a large number of overly large packets with the DF bit set is not desirable, but sending a large number of packets that are fragmented in the network can be equally undesirable.

If the sender doesn't know the Path MTU when the initial window is sent, the sender SHOULD send one large packet in the initial window with the DF bit set, and send the remaining packets in the initial window with a smaller MTU of 576 bytes (or 1280 bytes with IPv6).

A second possibility would be for the sender to delay sending the Quick-Start Request for one round-trip time by sending the Quick-Start Request with the first window of data, while also doing Path MTU Discovery.

The sender may be using an iterative approach such as Packetization Layer Path MTU Discovery (PLPMTUD) [MH06] for Path MTU Discovery, where the sender tests successively larger MTUs. If a probe is successfully delivered, then the MTU can be raised to reflect the value used in that probe. We would note that PLPMTUD does not allow the sender to determine the Path MTU before sending the initial window of data.

4.7.2. Quick-Start Request Packets that are Discarded by Routers or Middleboxes

It is always possible for a TCP SYN packet carrying a Quick-Start request to be dropped in the network due to congestion, or to be blocked due to interactions with routers or middleboxes, where a middlebox is defined as any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host [RFC3234]. Measurement studies of interactions between transport protocols and middleboxes [MAF04] show that for 70% of the Web servers investigated, no connection is established if the TCP SYN packet contains an unknown IP option (and for 43% of the Web servers, no connection is established if the TCP SYN packet contains an IP TimeStamp Option). In both cases, this is presumably due to routers or middleboxes along that path.

If the TCP sender doesn't receive a response to the SYN or SYN/ACK packet containing the Quick-Start Request, then the TCP sender SHOULD resend the SYN or SYN/ACK packet without the Quick-Start Request.

Similarly, if the TCP sender receives a TCP reset in response to the SYN or SYN/ACK packet containing the Quick-Start Request, then the TCP sender SHOULD resend the SYN or SYN/ACK packet without the Quick-Start Request [RFC3360].

RFCs 1122 and 2988 specify that the sender should set the initial RTO (retransmission timeout) to three seconds, though many TCP implementations set the initial RTO to one second. For a TCP SYN packet sent with a Quick-Start request, the TCP sender SHOULD use an initial RTO of three seconds.

We note that if the TCP SYN packet is using the IP Quick-Start Option for a Quick-Start Request, and it is also using bits in the TCP header to negotiate ECN-capability with the TCP host at the other end, then the drop of a TCP SYN packet could be due to congestion, a router or middlebox dropping the packet because of the IP Option, or a router or middlebox dropping the packet because of the information in the TCP header negotiating ECN. In this case, the sender could resend the dropped packet without either the Quick-Start or the ECN requests. Alternately, the sender could resend the dropped packet with only the ECN request in the TCP header, resending the TCP SYN packet without either the Quick-Start or the ECN requests if the second TCP SYN packet is dropped. The second choice seems reasonable, given that a TCP SYN packet today is more likely to be blocked due to policies that discard packets with IP Options than due to policies that discard packets with ECN requests in the TCP header [MAF04].

4.8. TCP: A Quick-Start Request in the Middle of a Connection

This section discusses the following issues that arise when Quick-Start is used by TCP to request a larger window in the middle of a connection, such as after an idle period: (1) determining the rate to request; (2) when to make a request; and (3) the response if Quick-Start packets are dropped.

(1) Determining the rate to request:
For a connection that has not yet had a congestion event (that is, a marked or dropped packet), the TCP sender is not restricted in the rate that it requests. As an example, a server might wait and send a Quick-Start Request after a short interaction with the client.

To use a Quick-Start Request in a connection that has already experienced a congestion event, and that has not had a recent mobility event, the TCP sender can determine the largest congestion window that the TCP connection achieved since the last packet drop and translate this to a sending rate to get the maximum allowed request rate. If the request is granted, then the sender essentially restarts with its old congestion window from before it was reduced, for example, during an idle period.

A Quick-Start Request sent in the middle of a TCP connection SHOULD be sent on a data packet.

(2) When to make a request:
A TCP connection MAY make a Quick-Start Request before the connection has experienced a congestion event, or after an idle period of at least one RTO.

(3) Response if Quick-Start packets are dropped:
If Quick-Start packets are dropped in the middle of connection, then the sender MUST revert to half the Quick-Start window, or to the congestion window that the sender would have used if the Quick-Start request had not been approved, whichever is smaller.

4.9. An Example Quick-Start Scenario with TCP

The following is an example scenario of when both hosts request Quick-Start for setting their initial windows. This is similar to Figures 1 and 2 in Section 2.1, except that it illustrates a TCP connection with both TCP hosts sending Quick-Start Requests.

* The TCP SYN packet from Host A contains a Quick-Start Request in the IP header.

* Routers along the forward path modify the Quick-Start Request as appropriate.

* Host B receives the Quick-Start Request in the SYN packet, and calculates the TTL Diff. If Host B approves the Quick-Start Request, then Host B sends a Quick-Start Response in the TCP header of the SYN/ACK packet. Host B also sends a Quick-Start Request in the IP header of the SYN/ACK packet.

* Routers along the reverse path modify the Quick-Start Request as appropriate.

* Host A receives the Quick-Start Response in the SYN/ACK packet, and checks the TTL Diff, Rate Request, and QS Nonce for validity. If they are valid, then Host A sets its initial congestion window appropriately, and sets up rate-based pacing to be used with the initial window. If the Quick-Start Response is not valid, then Host A uses TCP's default initial window. In either case, Host A sends a Report of Approved Rate.

Host A also calculates the TTL Diff for the Quick-Start Request in the incoming SYN/ACK packet, and sends a Quick-Start Response in the TCP header of the ACK packet.

* Host B receives the Quick-Start Response in an ACK packet, and checks the TTL Diff, Rate Request, and QS Nonce for validity. If the Quick-Start Response is valid, then Host B sets its initial congestion window appropriately, and sets up rate-based pacing to be used with its initial window. If the Quick-Start Response is not valid, then Host B uses TCP's default initial window. In either case, Host B sends a Report of Approved Rate.

5. Quick-Start and IPsec AH

This section shows that Quick-Start is compatible with IPsec Authentication Header (AH). AH uses an Integrity Check Value (ICV) in the IPsec Authentication Header to verify both message authentication and integrity [RFC4302]. Changes to the Quick-Start Option in the IP header do not affect this AH ICV. The tunnel considerations in Section 6 below apply to all IPsec tunnels, regardless of what IPsec headers or processing are used in conjunction with the tunnel.

Because the contents of the Quick-Start Option can change along the path, it is important that these changes not affect the IPsec Authentication Header Integrity Check Value (AH ICV). For IPv4, RFC 4302 requires that unrecognized IPv4 options be zeroed for AH ICV computation purposes, so Quick-Start IP Option data changing en route does not cause problems with existing IPsec AH implementations for IPv4. If the Quick-Start Option is recognized, it MUST be treated as a mutable IPv4 option, and hence be completely zeroed for AH ICV calculation purposes. IPv6 option numbers explicitly indicate whether the option is mutable; the third-highest order bit in the IANA-allocated option type has the value 1 to indicate that the Quick-Start Option data can change en route. RFC 4302 requires that the option data of any such option be zeroed for AH ICV computation purposes. Therefore, changes to the Quick-Start Option in the IP header do not affect the calculation of the AH ICV.

6. Quick-Start in IP Tunnels and MPLS

This section considers interactions between Quick-Start and IP tunnels, including IPsec ([RFC4301]), IP in IP [RFC2003], GRE [RFC2784], and others. This section also considers interactions between Quick-Start and MPLS [RFC3031].

In the discussion, we use TTL Diff, defined earlier as the difference between the IP TTL and the Quick-Start TTL, mod 256. Recall that the sender considers the Quick-Start Request approved only if the value of TTL Diff for the packet entering the network is the same as the value of TTL Diff for the packet exiting the network.

Simple tunnels: IP tunnel modes are generally based on adding a new "outer" IP header that encapsulates the original or "inner" IP header and its associated packet. In many cases, the new "outer" IP header may be added and removed at intermediate points along a path, enabling the network to establish a tunnel without requiring endpoint participation. We denote tunnels that specify that the outer header be discarded at tunnel egress as "simple tunnels", and we denote tunnels where the egress saves and uses information from the outer header before discarding it as "non-simple tunnels". An example of a "non-simple tunnel" would be a tunnel configured to support ECN, where the egress router might copy the ECN codepoint in the outer header to the inner header before discarding the outer header [RFC3168].

```
                  __ Tunnels Compatible with Quick-Start
                 /
Simple Tunnels  __/
                 \
                  \__ Tunnels Not Compatible with Quick-Start
                                (False Positives!)


                      __ Tunnels Supporting Quick-Start
                     /
                    /
Non-Simple Tunnels __/_____ Tunnels Compatible with Quick-Start,
                   \            but Not Supporting Quick-Start
                    \
                     \__ Tunnels Not Compatible with Quick-Start?
```

Figure 6: Categories of Tunnels.

Tunnels that are compatible with Quick-Start: We say that an IP tunnel 'is not compatible with Quick-Start' if the use of a Quick-Start Request over such a tunnel allows false positives, where the TCP sender incorrectly believes that the Quick-Start Request was approved by all routers along the path. If the use of Quick-Start over the tunnel does not cause false positives, we say that the IP tunnel 'is compatible with Quick-Start'.

If the IP TTL of the inner header is decremented during forwarding before tunnel encapsulation takes place, then the simple tunnel is compatible with Quick-Start, with Quick-Start Requests being rejected. Section 6.1 describes in more detail the ways that a simple tunnel can be compatible with Quick-Start.

There are some simple tunnels that are not compatible with Quick-Start, allowing 'false positives' where the TCP sender incorrectly believes that the Quick-Start Request was approved by all routers along the path. This is discussed in Section 6.2 below.

One of our tasks in the future will be to investigate the occurrence of tunnels that are not compatible with Quick-Start, and to track the extent to which such tunnels are modified over time. The evaluation of the problem of false positives from tunnels that are not compatible with Quick-Start will affect the progression of Quick-Start from Experimental to Proposed Standard, and will affect the degree of deployment of Quick-Start while in Experimental mode.

Tunnels that support Quick-Start: We say that an IP tunnel 'supports Quick-Start' if it allows routers along the tunnel path to process the Quick-Start Request and give feedback, resulting in the appropriate possible acceptance of the Quick-Start Request. Some tunnels that are compatible with Quick-Start support Quick-Start, while others do not. We note that a simple tunnel is not able to support Quick-Start.

From a security point of view, the use of Quick-Start in the outer header of an IP tunnel might raise security concerns because an adversary could tamper with the Quick-Start information that propagates beyond the tunnel endpoint, or because the Quick-Start Option exposes information to network scanners. Our approach is to make supporting Quick-Start an option for IP tunnels. That is, in environments or tunneling protocols where the risks of using Quick-Start are judged to outweigh its benefits, the tunnel can simply delete the Quick-Start Option or zero the Quick-Start rate request and QS TTL fields before encapsulation. The result is that there are two viable options for IP tunnels to be compatible with Quick-Start. The first option is the simple tunnel described above and in Section 6.1, where the tunnel is compatible with Quick-Start but does not support Quick-Start, where all Quick-Start Requests along the path will be rejected. The second approach is a Quick-Start-capable mode, described in Section 6.3, where the tunnel actively supports Quick-Start.

6.1. Simple Tunnels that Are Compatible with Quick-Start

This section describes the ways that a simple tunnel can be compatible with Quick-Start but not support Quick-Start, resulting in the rejection of all Quick-Start Requests that traverse the tunnel.

If the tunnel ingress for the simple tunnel is at a router, the IP TTL of the inner header is generally decremented during forwarding before tunnel encapsulation takes place. In this case, TTL Diff will be changed, correctly causing the Quick-Start Request to be rejected. For a simple tunnel, it is preferable if the Quick-Start Request is not copied to the outer header, saving the routers within the tunnel from unnecessarily processing the Quick-Start Request. However, the Quick-Start Request will be rejected correctly in this case whether or not the Quick-Start Request is copied to the outer header.

6.1.1. Simple Tunnels that Are Aware of Quick-Start

If a tunnel ingress is aware of Quick-Start, but does not want to support Quick-Start, then the tunnel ingress MUST either zero the Quick-Start rate request, QS TTL, and QS Nonce fields, or remove the Quick-Start Option from the inner header before encapsulation. Section 6.3 describes the procedures for a tunnel that does want to support Quick-Start.

Deleting the Quick-Start Option or zeroing the Quick-Start rate request *after decapsulation* also serves to prevent the propagation of Quick-Start information, and is compatible with Quick-Start. If the outer header does not contain a Quick-Start Request, a Quick-Start-aware tunnel egress MUST reject the inner Quick-Start Request by zeroing the Rate Request field in the inner header, or by deleting the Quick-Start Option.

If the tunnel ingress is at a sending host or router where the IP TTL is not decremented prior to encapsulation, and neither tunnel endpoint is aware of Quick-Start, then this allows false positives, described in the section below.

6.2. Simple Tunnels that Are Not Compatible with Quick-Start

Sometimes a tunnel implementation that does not support Quick-Start is independent of the TCP sender or a router implementation that supports Quick-Start. In these cases, it is possible that a Quick-Start Request gets erroneously approved without the routers in the tunnel having individually approved the request, causing a false positive.

If a tunnel ingress is a separate component from the TCP sender or IP forwarding, it is possible that a packet with a Quick-Start option is encapsulated without the IP TTL being decremented first, or with both IP TTL and QS TTL being decremented before the tunnel encapsulation takes place. If the tunnel ingress does not know about Quick-Start, a valid Quick-Start Request with unchanged TTL Diff traverses in the inner header, while the outer header most likely does not carry a Quick-Start Request. If the tunnel egress also does not support Quick-Start, it remains possible that the Quick-Start Request would be falsely approved, because the packet is decapsulated using the Quick-Start Request from the inner header, and the value of TTL Diff echoed to the sender remains unchanged. For example, such a scenario can occur with a Bump-In-The-Stack (BITS), an IPsec encryption implementation where the data encryption occurs between the network drivers and the TCP/IP protocol stack [RFC4301].

As one example, if a remote access VPN client uses a BITS structure, then Quick-Start obstacles between the client and the VPN gateway won't be seen. This is a particular problem because the path between the client and the VPN gateway is likely to contain the most congested part of the path. Because most VPN clients are reported to use BITS [H05], we will explore this in more detail.

A Bump-In-The-Wire (BITW) is an IPsec encryption implementation where the encryption occurs on an outboard processor, offloading the encryption processing overhead from the host or router [RFC4301]. The BITW device is usually IP addressable, which means that the IP TTL is decremented before the packet is passed to the BITW. If the QS TTL is not decremented, then the value of TTL Diff is changed, and the Quick-Start Request will be denied. However, if the BITW supports a host and does not have its own IP address, then the IP TTL is not decremented before the packet is passed from the host to the BITW, and a false positive could occur.

Other tunnels that need to be looked at are IP tunnels over non-network protocols, such as IP over TCP and IP over UDP [RFC3948], and tunnels using the Layer Two Tunneling Protocol [RFC2661].

Section 9.2 discusses the related issue of non-IP queues, such as layer-two Ethernet or ATM (Asynchronous Transfer Mode) networks, as another instance of possible bottlenecks that do not participate in the Quick-Start feedback.

6.3. Tunnels That Support Quick-Start

This section discusses tunnels configured to support Quick-Start.

If the tunnel ingress node chooses to locally approve the Quick-Start Request, then the ingress node MUST decrement the Quick-Start TTL at the same time it decrements the IP TTL, and MUST copy IP TTL and the Quick-Start Option from the inner IP header to the outer header. During encapsulation, the tunnel ingress MUST zero the Quick-Start rate request field in the inner header to ensure that the Quick-Start Request will be rejected if the tunnel egress does not support Quick-Start.

If the tunnel ingress node does not choose to locally approve the Quick-Start Request, then it MUST either delete the Quick-Start option from the inner header before encapsulation, or zero the QS TTL and the Rate Request fields before encapsulation.

Upon decapsulation, if the outer header contains a Quick-Start option, the tunnel egress MUST copy the IP TTL and the Quick-Start option from the outer IP header to the inner header.

IPsec uses the IKE (Internet Key Exchange) Protocol for security associations. We do not consider the interactions between Quick-Start and IPsec with IKEv1 [RFC2409] in this document. Now that the RFC for IKEv2 [RFC4306] is published, we plan to specify a modification of IPsec to allow the support of Quick-Start to be negotiated; this modification will specify the negotiation between tunnel endpoints to allow or forbid support for Quick-Start within the tunnel. This was done for ECN for IPsec tunnels, with IKEv1 [RFC3168, Section 9.2]. This negotiation of Quick-Start capability in an IPsec tunnel will be specified in a separate IPsec document. This document will also include a discussion of the potential effects of an adversary's modifications of the Quick-Start field (as in Sections 18 and 19 of RFC 3168), and of the security considerations of exposing the Quick-Start rate request to network scanners.

6.4. Quick-Start and MPLS

The behavior of Quick-Start with MPLS is similar to the behavior of Quick-Start with IP Tunnels. For those MPLS paths where the IP TTL is decremented as part of traversing the MPLS path, these paths are compatible with Quick-Start, but do not support Quick-Start; Quick-

Start Requests that are traversing these paths will be correctly understood by the transport sender as having been denied. Any MPLS paths where the IP TTL is not decremented as part of traversing the MPLS path would be not compatible with Quick-Start; such paths would result in false positives, where the TCP sender incorrectly believes that the Quick-Start Request was approved by all routers along the path.

For cases where the ingress node to the MPLS path is aware of Quick-Start, this node should either zero the Quick-Start rate request, QS TTL, and QS Nonce fields, or remove the Quick-Start Option from the IP header.

7. The Quick-Start Mechanism in Other Transport Protocols

The section earlier specified the use of Quick-Start in TCP. In this section, we generalize this to give guidelines for the use of Quick-Start with other transport protocols. We also discuss briefly how Quick-Start could be specified for other transport protocols.

The general guidelines for Quick-Start in transport protocols are as follows:

* Quick-Start is only specified for unicast transport protocols with appropriate congestion control mechanisms. Note: Quick-Start is not a replacement for standard congestion control techniques, but meant to augment their operation.

* A transport-level mechanism is needed for the Quick-Start Response from the receiver to the sender. This response contains the Rate Request, TTL Diff, and QS Nonce.

* The sender checks the validity of the Quick-Start Response.

* The sender has an estimate of the round-trip time, and translates the Quick-Start Response into an allowed window or allowed sending rate. The sender sends a Report of the Approved Rate. The sender starts sending Quick-Start packets, rate-paced out at the approved sending rate.

* After the sender receives the first acknowledgement packet for a Quick-Start packet, no more Quick-Start packets are sent. The sender adjusts its current congestion window or sending rate to be consistent with the actual amount of data that was transmitted in that round-trip time.

* When the last Quick-Start packet is acknowledged, the sender continues using the standard congestion control mechanisms of that protocol.

* If one of the Quick-Start packets is lost, then the sender reverts to the standard congestion control method of that protocol that would have been used if the Quick-Start Request had not been approved. In addition, the sender takes into account the information that the Quick-Start congestion window was too large (e.g., by decreasing ssthresh in TCP).

8. Using Quick-Start

8.1. Determining the Rate to Request

As discussed in [SAF06], the data sender does not necessarily have information about the size of the data transfer at connection initiation; for example, in request-response protocols such as HTTP, the server doesn't know the size or name of the requested object during connection initiation. [SAF06] explores some of the performance implications of overly large Quick-Start Requests, and discusses heuristics that end-nodes could use to size their requests appropriately. For example, the sender might have information about the bandwidth of the last-mile hop, the size of the local socket buffer, or of the TCP receive window, and could use this information in determining the rate to request. Web servers that mostly have small objects to transfer might decide not to use Quick-Start at all, since Quick-Start would be of little benefit to them.

Quick-Start will be more effective if Quick-Start Requests are not larger than necessary; every Quick-Start Request that is approved but not used (or not fully used) takes away from the bandwidth pool available for granting successive Quick-Start Requests.

8.2. Deciding the Permitted Rate Request at a Router

In this section, we briefly outline how a router might decide whether or not to approve a Quick-Start Request. The router should ask the following questions:

* Has the router's output link been underutilized for some time (e.g., several seconds)?

* Would the output link remain underutilized if the arrival rate were to increase by the aggregate rate requests that the router has approved over the last fraction of a second?

In order to answer the last question, the router must have some knowledge of the available bandwidth on the output link and of the Quick-Start bandwidth that could arrive due to recently approved Quick-Start Requests. In this way, if an underutilized router experiences a flood of Quick-Start Requests, the router can begin to deny Quick-Start Requests while the output link is still underutilized.

A simple way for the router to keep track of the potential bandwidth from recently approved requests is to maintain two counters: one for the total aggregate Rate Requests that have been approved in the current time interval [T1, T2], and one for the total aggregate Rate Requests approved over a previous time interval [T0, T1]. However, this document doesn't specify router algorithms for approving Quick-Start Requests, or make requirements for the appropriate time intervals for remembering the aggregate approved Quick-Start bandwidth. A possible router algorithm is given in Appendix E, and more discussion of these issues is available in [SAF06].

* If the router's output link has been underutilized and the aggregate of the Quick-Start Request Rate options granted is low enough to prevent a near-term bandwidth shortage, then the router could approve the Quick-Start Request.

Section 10.2 discusses some of the implementation issues in processing Quick-Start Requests at routers. [SAF06] discusses the range of possible Quick-Start algorithms at the router for deciding whether to approve a Quick-Start Request. In order to explore the limits of the possible functionality at routers, [SAF06] also discusses Extreme Quick-Start mechanisms at routers, where the router would keep per-flow state concerning approved Quick-Start requests.

9. Evaluation of Quick-Start

9.1. Benefits of Quick-Start

The main benefit of Quick-Start is the faster start-up for the transport connection itself. For a small TCP transfer of one to five packets, Quick-Start is probably of very little benefit; at best, it might shorten the connection lifetime from three to two round-trip times (including the round-trip time for connection establishment). Similarly, for a very large transfer, where the slow-start phase would have been only a small fraction of the connection lifetime, Quick-Start would be of limited benefit. Quick-Start would not significantly shorten the connection lifetime, but it might eliminate or at least shorten the start-up phase. However, for moderate-sized connections in a well-provisioned environment, Quick-Start could possibly allow the entire transfer of M packets to be completed in one round-trip time (after the initial round-trip time for the SYN exchange), instead of the log_2(M)-2 round-trip times that it would normally take for the data transfer, in an uncongested environments (assuming an initial window of four packets).

9.2. Costs of Quick-Start

This section discusses the costs of Quick-Start for the connection and for the routers along the path.

The cost of having a Quick-Start Request packet dropped:
Measurement studies cited earlier [MAF04] suggest that on a wide range of paths in the Internet, TCP SYN packets containing unknown IP options will be dropped. Thus, for the sender one risk in using Quick-Start is that the packet carrying the Quick-Start Request could be dropped in the network. It is particularly costly to the sender when a TCP SYN packet is dropped, because in this case the sender should wait for an RTO of three seconds before re-sending the SYN packet, as specified in Section 4.7.2.

The cost of having a Quick-Start data packet dropped:
Another risk for the sender in using Quick-Start lies in the possibility of suffering from congestion-related losses of the Quick-Start data packets. This should be an unlikely situation because routers are expected to approve Quick-Start Requests only when they are significantly underutilized. However, a transient increase in cross-traffic in one of the routers, a sudden decrease in available bandwidth on one of the links, or congestion at a non-IP queue could result in packet losses even when the Quick-Start Request was approved by all of the routers along the path. If a Quick-Start packet is dropped, then the sender reverts to the congestion control mechanisms it would have used if the Quick-Start Request had not been approved, so the performance cost to the connection of having a Quick-Start packet dropped is small, compared to the performance without Quick-Start. (On the other hand, the performance difference between Quick-Start with a Quick-Start packet dropped and Quick-Start with no Quick-Start packet dropped can be considerable.)

Added complexity at routers:
The main cost of Quick-Start at routers concerns the costs of added complexity. The added complexity at the end-points is moderate, and might easily be outweighed by the benefit of Quick-Start to the end hosts. The added complexity at the routers is also somewhat moderate; it involves estimating the unused bandwidth on the output link over the last several seconds, processing the Quick-Start request, and keeping a counter of the aggregate Quick-Start rate approved over the last fraction of a second. However, this added complexity at routers adds to the development cycle, and could

prevent the addition of other competing functionality to routers. Thus, careful thought would have to be given to the addition of Quick-Start to IP.

The slow path in routers:
Another drawback of Quick-Start is that packets containing the Quick-Start Request message might not take the fast path in routers, particularly in the beginning of Quick-Start's deployment in the Internet. This would mean some extra delay for the end hosts, and extra processing burden for the routers. However, as discussed in Sections 4.1 and 4.7, not all packets would carry the Quick-Start option. In addition, for the underutilized links where Quick-Start Requests could actually be approved, or in typical environments where most of the packets belong to large flows, the burden of the Quick-Start Option on routers would be considerably reduced. Nevertheless, it is still conceivable, in the worst case, that many packets would carry Quick-Start Requests; this could slow down the processing of Quick-Start packets in routers considerably. As discussed in Section 9.6, routers can easily protect against this by enforcing a limit on the rate at which Quick-Start Requests will be considered. [RW03] and [RW04] contain measurements of the impact of IP Option Processing on packet round-trip times.

Multiple paths:

One limitation of Quick-Start is that it presumes that the data packets of a connection will follow the same path as the Quick-Start request packet. If this is not the case, then the connection could be sending the Quick-Start packets, at the approved rate, along a path that was already congested, or that became congested as a result of this connection. Thus, Quick-Start could give poor performance when there is a routing change immediately after the Quick-Start Request is approved, and the Quick-Start data packets follow a different path from that of the original Quick-Start Request. This is, however, similar to what would happen for a connection with sufficient data, if the connection's path was changed in the middle of the connection, which had already established the allowed initial rate.

As specified in Section 3.3, a router that uses multipath routing for packets within a single connection must not approve a Quick-Start Request. Quick-Start would not perform robustly in an environment with multipath routing, where different packets in a connection routinely follow different paths. In such an environment, the Quick-Start Request and some fraction of the packets in the connection might take an underutilized path, while the rest of the packets take an alternate, congested path.

Non-IP queues:
A problem of any mechanism for feedback from routers at the IP level is that there can be queues and bottlenecks in the end-to-end path that are not in IP-level routers. As an example, these include queues in layer-two Ethernet or ATM networks. One possibility would be that an IP-level router adjacent to such a non-IP queue or bottleneck would be configured to reject Quick-Start Requests if that was appropriate. One would hope that, in general, IP networks are configured so that non-IP queues between IP routers do not end up being the congested bottlenecks.

9.3. Quick-Start with QoS-Enabled Traffic

The discussion in this document has largely been of Quick-Start with default, best-effort traffic. However, Quick-Start could also be used by traffic using some form of differentiated services, and routers could take the traffic class into account when deciding whether or not to grant the Quick-Start Request. We don't address this context further in this paper, since it is orthogonal to the specification of Quick-Start.

Routers are also free to take into account their own priority classifications in processing Quick-Start Requests.

9.4. Protection against Misbehaving Nodes

In this section, we discuss the protection against senders, receivers, or colluding routers or middleboxes lying about the Quick-Start Request.

9.4.1. Misbehaving Senders

A transport sender could try to transmit data at a higher rate than that approved in the Quick-Start Request. The network could use a traffic policer to protect against misbehaving senders that exceed the approved rate, for example, by dropping packets that exceed the allowed transmission rate. The required Report of Approved Rate allows traffic policers to check that the Report of Approved Rate does not exceed the Rate Request actually approved at that point in the network in the previous Quick-Start Request from that connection. The required Approved Rate report also allows traffic policers to check that the sender's sending rate does not exceed the rate in the Report of Approved Rate.

If a router or receiver receives an Approved Rate report that is larger than the Rate Request in the Quick-Start Request approved for that sender for that connection in the previous round-trip time, then the router or receiver could deny future Quick-Start Requests from that sender, e.g., by deleting the Quick-Start Request from future packets from that sender. We note that routers are not required to use Approved Rate reports to check if senders are cheating; this is at the discretion of the router.

If a router sees a Report of Approved Rate, and did not see an earlier Quick-Start Request, then either the sender could be cheating, or the connection's path could have changed since the Quick-Start Request was sent. In either case, the router could decide to deny future Quick-Start Requests for this connection. In particular, it is reasonable for the router to deny a Quick-Start request if either the sender is cheating, or if the connection path suffers from path changes or multipathing.

If a router approved a Quick-Start Request, but does not see a subsequent Approved Rate report, then there are several possibilities: (1) the request was denied and/or dropped downstream, and the sender did not send a Report of Approved Rate; (2) the request was approved, but the sender did not send a Report of Approved Rate; (3) the Approved Rate report was dropped in the network; or (4) the Approved Rate report took a different path from the Quick-Start Request. In any of these cases, the router would be justified in denying future Quick-Start Requests for this connection.

In any of the cases mentioned in the three paragraphs above (i.e., an Approved Rate report that is larger than the Rate Request in the earlier Quick-Start Request, a Report of Approved Rate with no preceding Rate Request, or a Rate Request with no Report of Approved Rate), a traffic policer may assume that Quick-Start is not being used appropriately, or is being used in an unsuitable environment (e.g., with multiple paths), and take some corresponding action.

What are the incentives for a sender to cheat by over-sending after a Quick-Start Request? Assuming that the sender's interests are measured by a performance metric such as the completion time for its connections, sometimes it might be in the sender's interests to cheat, and sometimes it might not; in some cases, it could be difficult for the sender to judge whether it would be in its interests to cheat. The incentives for a sender to cheat by over-sending after a Quick-Start Request are not that different from the incentives for a sender to cheat by over-sending even in the absence of Quick-Start, with one difference: the use of Quick-Start could help a sender evade policing actions from policers in the network. The Report of Approved Rate is designed to address this and to make it harder for senders to use Quick-Start to 'cover' their cheating.

9.4.2. Receivers Lying about Whether the Request was Approved

One form of misbehavior would be for the receiver to lie to the sender about whether the Quick-Start Request was approved, by falsely reporting the TTL Diff and QS Nonce. If a router that understands the Quick-Start Request denies the request by deleting the request or by zeroing the QS TTL and QS Nonce, then the receiver can "lie" about whether the request was approved only by successfully guessing the value of the TTL Diff and QS Nonce to report. The chance of the receiver successfully guessing the correct value for the TTL Diff is 1/256, and the chance of the receiver successfully guessing the QS nonce for a reported rate request of K is 1/(2K).

However, if the Quick-Start Request is denied only by a non-Quick-Start-capable router, or by a router that is unable to zero the QS TTL and QS Nonce fields, then the receiver could lie about whether the Quick-Start Requests were approved by modifying the QS TTL in successive requests received from the same host. In particular, if the sender does not act on a Quick-Start Request, then the receiver could decrement the QS TTL by one in the next request received from that host before calculating the TTL Diff, and decrement the QS TTL by two in the following received request, until the sender acts on one of the Quick-Start Requests.

Unfortunately, if a router doesn't understand Quick-Start, then it is not possible for that router to take an active step such as zeroing the QS TTL and QS Nonce to deny a request. As a result, the QS TTL is not a fail-safe mechanism for preventing lying by receivers in the case of non-Quick-Start-capable routers.

What would be the incentives for a receiver to cheat in reporting on a Quick-Start Request, in the absence of a mechanism such as the QS Nonce? In some cases, cheating would be of clear benefit to the receiver, resulting in a faster completion time for the transfer. In other cases, where cheating would result in Quick-Start packets being dropped in the network, cheating might or might not improve the receiver's performance metric, depending on the details of that particular scenario.

9.4.3. Receivers Lying about the Approved Rate

A second form of receiver misbehavior would be for the receiver to lie to the sender about the Rate Request for an approved Quick-Start Request, by increasing the value of the Rate Request field. However, the receiver doesn't necessarily know the Rate Request in the original Quick-Start Request sent by the sender, and a higher Rate Request reported by the receiver will only be considered valid by the sender if it is no higher than the Rate Request originally requested by the sender. For example, if the sender sends a Quick-Start Request with a Rate Request of X, and the receiver reports receiving a Quick-Start Request with a Rate Request of Y > X, then the sender knows that either some router along the path malfunctioned (increasing the Rate Request inappropriately), or the receiver is lying about the Rate Request in the received packet.

If the sender sends a Quick-Start Request with a Rate Request of Z, the receiver receives the Quick-Start Request with an approved Rate Request of X, and reports a Rate Request of Y, for X < Y <= Z, then the receiver only succeeds in lying to the sender about the approved rate if the receiver successfully reports the rightmost 2Y bits in the QS nonce.

If senders often use a configured default value for the Rate Request, then receivers would often be able to guess the original Rate Request, and this would make it easier for the receiver to lie about the value of the Rate Request field. Similarly, if the receiver often communicates with a particular sender, and the sender always uses the same Rate Request for that receiver, then the receiver might over time be able to infer the original Rate Request used by the sender.

There are several possible additional forms of protection against receivers lying about the value of the Rate Request. One possible additional protection would be for a router that decreases a Rate Request in a Quick-Start Request to report the decrease directly to the sender. However, this could lead to many reports back to the sender for a single request, and could also be used in address-spoofing attacks.

A second limited form of protection would be for senders to use some degree of randomization in the requested Rate Request, so that it is difficult for receivers to guess the original value for the Rate Request. However, this is difficult because there is a fairly coarse granularity in the set of rate requests available to the sender, and randomizing the initial request only offers limited protection, in any case.

9.4.4. Collusion between Misbehaving Routers

In addition to protecting against misbehaving receivers, it is necessary to protect against misbehaving routers. Consider collusion between an ingress router and an egress router belonging to the same intranet. The ingress router could decrement the Rate Request at the ingress, with the egress router increasing it again at the egress. The routers between the ingress and egress that approved the decremented rate request might not have been willing to approve the larger, original request.

Another form of collusion would be for the ingress router to inform the egress router out-of-band of the TTL Diff and QS Nonce for the request packet at the ingress. This would enable the egress router to modify the QS TTL and QS Nonce so that it appeared that all the routers along the path had approved the request. There does not appear to be any protection against a colluding ingress and egress router. Even if an intermediate router had deleted the Quick-Start Option from the packet, the ingress router could have sent the Quick-Start Option to the egress router out-of-band, with the egress router inserting the Quick-Start Option, with a modified QS TTL field, back in the packet.

However, unlike ECN, there is somewhat less of an incentive for cooperating ingress and egress routers to collude to falsely modify the Quick-Start Request so that it appears to have been approved by all the routers along the path. With ECN, a colluding ingress router could falsely mark a packet as ECN-capable, with the colluding egress router returning the ECN field in the IP header to its original non-ECN-capable codepoint, and congested routers along the path could have been fooled into not dropping that packet. This collusion would give an unfair competitive advantage to the traffic protected by the colluding ingress and egress routers.

In contrast, with Quick-Start, the collusion of the ingress and egress routers to make it falsely appear that a Quick-Start Request was approved sometimes would give an advantage to the traffic covered by that collusion, and sometimes would give a disadvantage, depending on the details of the scenario. If some router along the path really does not have enough available bandwidth to approve the Quick-Start Request, then Quick-Start packets sent as a result of the falsely approved request could be dropped in the network, to the possible disadvantage of the connection. Thus, while the ingress and egress routers could collude to prevent intermediate routers from denying a Quick-Start Request, it would not always be to the connection's advantage for this to happen. One defense against such a collusion would be for some router between the ingress and egress nodes that denied the request to monitor connection performance, penalizing connections that seem to be using Quick-Start after a Quick-Start Request was denied, or that are reporting an Approved Rate higher than that actually approved by that router.

If the congested router is ECN-capable, and the colluding ingress and egress routers are lying about ECN-capability as well as about Quick-Start, then the result could be that the Quick-Start Request falsely appears to the sender to have been approved, and the Quick-

Start packets falsely appear to the congested router to be ECN-capable. In this case, the colluding routers might succeed in giving a competitive advantage to the traffic protected by their collusion (if no intermediate router is monitoring to catch such misbehavior).

9.5. Misbehaving Middleboxes and the IP TTL

One possible difficulty is that of traffic normalizers [HKP01], or other middleboxes along that path, that rewrite IP TTLs in order to foil other kinds of attacks in the network. If such a traffic normalizer rewrote the IP TTL, but did not adjust the Quick-Start TTL by the same amount, then the sender's mechanism for determining if the request was approved by all routers along the path would no longer be reliable. Rewriting the IP TTL could result in false positives (with the sender incorrectly believing that the Quick-Start Request was approved) as well as false negatives (with the sender incorrectly believing that the Quick-Start Request was denied).

9.6. Attacks on Quick-Start

As discussed in [SAF06], Quick-Start is vulnerable to two kinds of attacks: (1) attacks to increase the routers' processing and state load and (2) attacks with bogus Quick-Start Requests to temporarily tie up available Quick-Start bandwidth, preventing routers from approving Quick-Start Requests from other connections. Routers can protect against the first kind of attack by applying a simple limit on the rate at which Quick-Start Requests will be considered by the router.

The second kind of attack, to tie up the available Quick-Start bandwidth, is more difficult to defend against. As discussed in [SAF06], Quick-Start Requests that are not going to be used, either because they are from malicious attackers or because they are denied by routers downstream, can result in short-term 'wasting' of potential Quick-Start bandwidth, resulting in routers denying subsequent Quick-Start Requests that, if approved, would in fact have been used.

We note that the likelihood of malicious attacks would be minimized significantly when Quick-Start was deployed in a controlled environment such as an intranet, where there was some form of centralized control over the users in the system. We also note that this form of attack could potentially make Quick-Start unusable, but it would not do any further damage; in the worst case, the network would function as a network without Quick-Start.

[SAF06] considers the potential of Extreme Quick-Start algorithms at routers, which keep per-flow state for Quick-Start connections, in protecting the availability of Quick-Start bandwidth in the face of frequent, overly large Quick-Start Requests.

9.7. Simulations with Quick-Start

Quick-Start was added to the NS simulator [SH02] by Srikanth Sundarrajan, and additional functionality was added by Pasi Sarolahti. The validation test is at 'test-all-quickstart' in the 'tcl/test' directory in NS. The initial simulation studies from [SH02] show a significant performance improvement using Quick-Start for moderate-sized flows (between 4 KB and 128 KB) in underutilized environments. These studies are of file transfers, with the improvement measured as the relative increase in the overall throughput for the file transfer. The study shows that potential improvement from Quick-Start is proportional to the delay-bandwidth product of the path.

The Quick-Start simulations in [SAF06] explore the following: the potential benefit of Quick-Start for the connection, the relative benefits of different router-based algorithms for approving Quick-Start Requests, and the effectiveness of Quick-Start as a function of the senders' algorithms for choosing the size of the rate request.

10. Implementation and Deployment Issues

This section discusses some of the implementation issues with Quick-Start. This section also discusses some of the key deployment issues, such as the chicken-and-egg deployment problems of mechanisms that have to be deployed in both routers and end nodes in order to work, and the problems posed by the wide deployment of middleboxes today that block the use of known or unknown IP Options.

10.1. Implementation Issues for Sending Quick-Start Requests

Section 4.7 discusses some of the issues with deciding the initial sending rate to request. Quick-Start raises additional issues about the communication between the transport protocol and the application, and about the use of past history with Quick-Start in the end node.

One possibility is that a protocol implementation could provide an API for applications to indicate when they want to request Quick-Start, and what rate they would like to request. In the conventional socket API, this could be a socket option that is set before a connection is established. Some applications, such as those that use TCP for bulk transfers, do not have interest in the transmission rate, but they might know the amount of data that can be sent immediately. Based on this, the sender implementation could decide whether Quick-Start would be useful, and what rate should be requested.

We note that when Quick-Start is used, the TCP sender is required to save the QS Nonce and the TTL Diff when the Quick-Start Request is sent, and to implement an additional timer for the paced transmission of Quick-Start packets.

10.2. Implementation Issues for Processing Quick-Start Requests

A router or other network host must be able to determine the approximate bandwidth of its outbound network interfaces in order to process incoming Quick-Start rate requests, including those that originate from the host itself. One possibility would be for hosts to rely on configuration information to determine link bandwidths; this has the drawback of not being robust to errors in configuration. Another possibility would be for network device drivers to infer the bandwidth for the interface and to communicate this to the IP layer.

Particular issues will arise for wireless links with variable bandwidth, where decisions will have to be made about how frequently the host gets updates of the changing bandwidth. It seems appropriate that Quick-Start Requests would be handled particularly conservatively for links with variable bandwidth; to avoid cases where Quick-Start Requests are approved, the link bandwidth is reduced, and the data packets that are sent end up being dropped.

Difficult issues also arise for paths with multi-access links (e.g., Ethernet). Routers or end-nodes with multi-access links should be particularly conservative in granting Quick-Start Requests. In particular, for some multi-access links, there may be no procedure for an attached node to use to determine whether all parts of the multi-access link have been underutilized in the recent past.

10.3. Possible Deployment Scenarios

Because of possible problems discussed above concerning using Quick-Start over some network paths and the security issues discussed in Section 11, the most realistic initial deployment of Quick-Start would most likely take place in intranets and other controlled environments. Quick-Start is most useful on high bandwidth-delay paths that are significantly underutilized. The primary initial users of Quick-Start would likely be in organizations that provide network services to their users and also have control over a large portion of the network path.

Quick-Start is not currently intended for ubiquitous deployment in the global Internet. In particular, Quick-Start should not be enabled by default in end-nodes or in routers; instead, when Quick-Start is used, it should be explicitly enabled by users or system administrators.

Below are a few examples of networking environments where Quick-Start would potentially be useful. These are the environments that might consider an initial deployment of Quick-Start in the routers and end-nodes, where the incentives for routers to deploy Quick-Start might be the most clear.

* Centrally administrated organizational intranets: These intranets often have large network capacity, with networks that are underutilized for much of the time [PABL+05]. Such intranets might also include high-bandwidth and high-delay paths to remote sites. In such an environment, Quick-Start would be of benefit to users, and there would be a clear incentive for the deployment of Quick-Start in routers. For example, Quick-Start could be quite useful in high-bandwidth networks used for scientific computing.

* Wireless networks: Quick-Start could also be useful in high-delay environments of Cellular Wide-Area Wireless Networks, such as the GPRS [BW97] and their enhancements and next generations. For example, GPRS EDGE (Enhanced Data for GSM Evolution) is expected to provide wireless bandwidth of up to 384 Kbps (roughly 32 1500-byte packets per second) while the GPRS round-trip times range typically from a few hundred milliseconds to over a second, excluding any possible queueing delays in the network [GPAR02]. In addition, these networks sometimes have variable additional delays due to resource allocation that could be avoided by keeping the connection path constantly utilized, starting from initial slow-start. Thus, Quick-Start could be of significant benefit to users in these environments.

* Paths over satellite links: Geostationary Orbit (GEO) satellite links have one-way propagation delays on the order of 250 ms while the bandwidth can be measured in megabits per second [RFC2488]. Because of the considerable bandwidth-delay product on the link, TCP's slow-start is a major performance limitation in the beginning of the connection. A large initial congestion window would be useful to users of such satellite links.

* Single-hop paths: Quick-Start should work well over point-to-point single-hop paths, e.g., from a host to an adjacent server. Quick-Start would work over a single-hop IP path consisting of a multi-access link only if the host was able to determine if the path to the next IP hop has been significantly underutilized over the

recent past. If the multi-access link includes a layer-2 switch, then the attached host cannot necessarily determine the status of the other links in the layer-2 network.

10.4. A Comparison with the Deployment Problems of ECN

Given the glacially slow rate of deployment of ECN in the Internet to date [MAF05], it is disconcerting to note that some of the deployment problems of Quick-Start are even greater than those of ECN. First, unlike ECN, which can be of benefit even if it is only deployed on one of the routers along the end-to-end path, a connection's use of Quick-Start requires Quick-Start deployment on all of the routers along the end-to-end path. Second, unlike ECN, which uses an allocated field in the IP header, Quick-Start requires the extra complications of an IP Option, which can be difficult to pass through the current Internet [MAF05].

However, in spite of these issues, there is some hope for the deployment of Quick-Start, at least in protected corners of the Internet, because the potential benefits of Quick-Start to the user are considerably more dramatic than those of ECN. Rather than simply replacing the occasional dropped packet by an ECN-marked packet, Quick-Start is capable of dramatically increasing the throughput of connections in underutilized environments [SAF06].

11. Security Considerations

Sections 9.4 and 9.6 discuss the security considerations related to Quick-Start. Section 9.4 discusses the potential abuse of Quick-Start by senders or receivers lying about whether the request was approved or about the approved rate, and of routers in collusion to misuse Quick-Start. Section 9.5 discusses potential problems with traffic normalizers that rewrite IP TTLs in packet headers. All these problems could result in the sender using a Rate Request that was inappropriately large, or thinking that a request was approved when it was in fact denied by at least one router along the path. This inappropriate use of Quick-Start could result in congestion and an unacceptable level of packet drops along the path. Such congestion could also be part of a Denial of Service attack.

Section 9.6 discusses a potential attack on the routers' processing and state load from an attack of Quick-Start Requests. Section 9.6 also discusses a potential attack on the available Quick-Start bandwidth by sending bogus Quick-Start Requests for bandwidth that will not, in fact, be used. While this impacts the global usability of Quick-Start, it does not endanger the network as a whole since TCP uses standard congestion control if Quick-Start is not available.

Section 4.7.2 discusses the potential problem of packets with Quick-Start Requests dropped by middleboxes along the path.

As discussed in Section 5, for IPv4 IPsec Authentication Header Integrity Check Value (AH ICV) calculation, the Quick-Start Option is a mutable IPv4 option and hence completely zeroed for AH ICV calculation purposes. This is also the treatment required by RFC 4302 for unrecognized IPv4 options. The IPv6 Quick-Start Option's IANA-allocated option type indicates that it is a mutable option; hence, according to RFC 4302, its option data is required to be zeroed for AH ICV computation purposes. See RFC 4302 for further explanation.

Section 6.2 discusses possible problems of Quick-Start used by connections carried over simple tunnels that are not compatible with Quick-Start. In this case, it is possible that a Quick-Start Request is erroneously considered approved by the sender without the routers in the tunnel having individually approved the request, causing a false positive.

We note two high-order points here. First, the Quick-Start Nonce goes a long way towards preventing large-scale cheating. Second, even if a host occasionally uses Quick-Start when it is not approved by the entire network path, the network will not collapse. Quick-Start does not remove TCP's basic congestion control mechanisms; these will kick in when the network is heavily loaded, relegating any Quick-Start mistake to a transient.

12. IANA Considerations

Quick-Start requires an IP Option and a TCP Option.

12.1. IP Option

Quick-Start requires both an IPv4 Option Number (Section 3.1) and an IPv6 Option Number (Section 3.2).

IPv4 Option Number:

```
Copy Class Number Value Name
---- ----- ------ ----- ----
   0    00     25    25   QS    - Quick-Start
```

IPv6 Option Number [RFC2460]:

```
HEX         act  chg  rest
---         ---  ---  -----
  6          00   1   00110     Quick-Start
```

For the IPv6 Option Number, the first two bits indicate that the IPv6 node may skip over this option and continue processing the header if it doesn't recognize the option type, and the third bit indicates that the Option Data may change en-route.

In both cases, this document should be listed as the reference document.

12.2. TCP Option

Quick-Start requires a TCP Option Number (Section 4.2).

TCP Option Number:

```
Kind Length Meaning
---- ------ ------------------------------
  27 8      Quick-Start Response
```

This document should be listed as the reference document.

13. Conclusions

We are presenting the Quick-Start mechanism as a simple, understandable, and incrementally deployable mechanism that would be sufficient to allow some connections to start up with large initial rates, or large initial congestion windows, in over-provisioned, high-bandwidth environments. We expect there will be an increasing number of over-provisioned, high-bandwidth environments where the Quick-Start mechanism, or another mechanism of similar power, could be of significant benefit to a wide range of traffic. We are presenting the Quick-Start mechanism as a request for the community to provide feedback and experimentation on issues relating to Quick-Start.

14. Acknowledgements

The authors wish to thank Mark Handley for discussions of these issues. The authors also thank the End-to-End Research Group, the Transport Services Working Group, and members of IPAM's program on Large-Scale Communication Networks for both positive and negative feedback on this proposal. We thank Srikanth Sundarrajan for the initial implementation of Quick-Start in the NS simulator, and for the initial simulation study. Many thanks to David Black and Joe Touch for extensive feedback on Quick-Start and IP tunnels. We also thank Mohammed Ashraf, John Border, Bob Briscoe, Martin Duke, Tom Dunigan, Mitchell Erblich, Gorry Fairhurst, John Heidemann, Paul Hyder, Dina Katabi, and Vern Paxson for feedback. Thanks also to Gorry Fairhurst for the suggestion of adding the QS Nonce to the Report of Approved Rate.

The version of the QS Nonce in this document is based on a proposal from Guohan Lu [L05]. Earlier versions of this document contained an eight-bit QS Nonce, and subsequent versions discussed the possibility of a four-bit QS Nonce.

This document builds upon the concepts described in [RFC3390], [AHO98], [RFC2415], and [RFC3168]. Some of the text on Quick-Start in tunnels was borrowed directly from RFC 3168.

This document is the development of a proposal originally by Amit Jain for Initial Window Discovery.

Appendix A. Related Work

The Quick-Start proposal, taken together with HighSpeed TCP [RFC3649] or other transport protocols for high-bandwidth transfers, could go a significant way towards extending the range of performance for best-effort traffic in the Internet. However, there are many things that the Quick-Start proposal would not accomplish. Quick-Start is not a congestion control mechanism, and would not help in making more precise use of the available bandwidth -- that is, of achieving the goal of high throughput with low delay and low packet-loss rates. Quick-Start would not give routers more control over the decrease rates of active connections.

In addition, any evaluation of Quick-Start must include a discussion of the relative benefits of approaches that use no explicit information from routers, and of approaches that use more fine-grained feedback from routers as part of a larger congestion control mechanism. We discuss several classes of proposals in the sections below.

A.1. Fast Start-Ups without Explicit Information from Routers

One possibility would be for senders to use information from the packet streams to learn about the available bandwidth, without explicit information from routers. These techniques would not allow a start-up as fast as that available from Quick-Start in an underutilized environment; one already has to have sent some packets in order to use the packet stream to learn about available bandwidth. However, these techniques could allow a start-up considerably faster than the current Slow-Start. While it seems clear that approaches *without* explicit feedback from the routers will be strictly less powerful than is possible *with* explicit feedback, it is also possible that approaches that are more aggressive than Slow-Start are possible without the complexity involved in obtaining explicit feedback from routers.

Periodic packet streams:
[JD02] explores the use of periodic packet streams to estimate the available bandwidth along a path. The idea is that the one-way delays of a periodic packet stream show an increasing trend when the stream's rate is higher than the available bandwidth (due to an increasing queue). While [JD02] states that the proposed mechanism does not cause significant increases in network utilization, losses, or delays when done by one flow at a time, the approach could be problematic if conducted concurrently by a number of flows. [JD02] also gives an overview of some of the earlier work on inferring the available bandwidth from packet trains.

Swift-Start:
The Swift Start proposal from [PRAKS02] combines packet-pair and packet-pacing techniques. An initial congestion window of four segments is used to estimate the available bandwidth along the path. This estimate is then used to dramatically increase the congestion window during the second RTT of data transmission.

SPAND:
In the TCP/SPAND proposal from [ZQK00] for speeding up short data transfers, network performance information would be shared among many co-located hosts to estimate each connection's fair share of the network resources. Based on such estimation and the transfer size, the TCP sender would determine the optimal initial congestion window size. The design for TCP/SPAND uses a performance gateway that monitors all traffic entering and leaving an organization's network.

Sharing information among TCP connections:
The Congestion Manager [RFC3124] and TCP control block sharing [RFC2140] both propose sharing congestion information among multiple TCP connections with the same endpoints. With the Congestion Manager, a new TCP connection could start with a high initial cwnd, if it was sharing the path and the cwnd with a pre-existing TCP connection to the same destination that had already obtained a high congestion window. RFC 2140 discusses ensemble sharing, where an established connection's congestion window could be 'divided up' to be shared with a new connection to the same host. However, neither of these approaches addresses the case of a connection to a new destination, with no existing or recent connection (and therefore congestion control state) to that destination.

While continued research on the limits of the ability of TCP and other transport protocols to learn of available bandwidth without explicit feedback from the router seems useful, we note that there are several fundamental advantages of explicit feedback from routers.

(1) Explicit feedback is faster than implicit feedback:
One advantage of explicit feedback from the routers is that it allows the transport sender to reliably learn of available bandwidth in one round-trip time.

(2) Explicit feedback is more reliable than implicit feedback:
Techniques that attempt to assess the available bandwidth at connection start-up using implicit techniques are more error-prone than techniques that involve every element in the network path. While explicit information from the network can be wrong, it has a much better chance of being appropriate than an end-host trying to *estimate* an appropriate sending rate using "block box" probing techniques of the entire path.

A.2. Optimistic Sending without Explicit Information from Routers

Another possibility that has been suggested [S02] is for the sender to start with a large initial window without explicit permission from the routers and without bandwidth estimation techniques and for the first packet of the initial window to contain information, such as the size or sending rate of the initial window. The proposal would be that congested routers would use this information in the first data packet to drop or delay many or all of the packets from that initial window. In this way, a flow's optimistically large initial window would not force the router to drop packets from competing flows in the network. Such an approach would seem to require some mechanism for the sender to ensure that the routers along the path understood the mechanism for marking the first packet of a large initial window.

Obviously, there would be a number of questions to consider about an approach of optimistic sending.

(1) Incremental deployment:
One question would be the potential complications of incremental deployment, where some of the routers along the path might not understand the packet information describing the initial window.

(2) Congestion collapse:
There could also be concerns about congestion collapse if many flows used large initial windows, many packets were dropped from optimistic initial windows, and many congested links ended up carrying packets that are only going to be dropped downstream.

(3) Distributed Denial of Service attacks:
A third question would be the potential role of optimistic senders in amplifying the damage done by a Distributed Denial of Service (DDoS) attack (assuming attackers use compliant congestion control in the hopes of "flying under the radar").

(4) Performance hits if a packet is dropped:
A fourth issue would be to quantify the performance hit to the connection when a packet is dropped from one of the initial windows.

A.3. Fast Start-Ups with Other Information from Routers

There have been several proposals somewhat similar to Quick-Start, where the transport protocol collects explicit information from the routers along the path.

An IP Option about the free buffer size:
In related work, [P00] investigates the use of a slightly different IP option for TCP connections to discover the available bandwidth along the path. In that proposal, the IP option would query the routers along the path about the smallest available free buffer size. Also, the IP option would have been sent after the initial SYN exchange, when the TCP sender already had an estimate of the round-trip time.

The Performance Transparency Protocol:
The Performance Transparency Protocol (PTP) includes a proposal for a single PTP packet that would collect information from routers along the path from the sender to the receiver [W00]. For example, a single PTP packet could be used to determine the bottleneck bandwidth along a path.

ETEN:
Additional proposals for end nodes to collect explicit information from routers include one variant of Explicit Transport Error Notification (ETEN), which includes a cumulative mechanism to notify endpoints of aggregate congestion statistics along the path [KAPS02]. (A second variant in [KSEPA04] does not depend on cumulative congestion statistics from the network.)

A.4. Fast Start-Ups with more Fine-Grained Feedback from Routers

Proposals for more fine-grained, congestion-related feedback from routers include XCP [KHR02], MaxNet [MaxNet], and AntiECN marking [K03]. Appendix B.6 discusses in more detail the relationship between Quick-Start and proposals for more fine-grained per-packet feedback from routers.

XCP:
Proposals, such as XCP for new congestion control mechanisms based on more feedback from routers, are more powerful than Quick-Start, but also are more complex to understand and more difficult to deploy. XCP routers maintain no per-flow state, but provide more fine-grained feedback to end-nodes than the one-bit congestion feedback of ECN. The per-packet feedback from XCP can be positive or negative, and specifies the increase or decrease in the sender's congestion window when this packet is acknowledged. XCP is a full-fledge congestion control scheme, whereas Quick-Start represents a quick check to determine if the network path is significantly underutilized such that a connection can start faster and then fall back to TCP's standard congestion control algorithms.

AntiECN:
The AntiECN proposal is for a single bit in the packet header that routers could set to indicate that they are underutilized. For each TCP ACK arriving at the sender indicating that a packet has been received with the Anti-ECN bit set, the sender would be able to increase its congestion window by one packet, as it would during slow-start.

A.5. Fast Start-Ups with Lower-Than-Best-Effort Service

There have been proposals for routers to provide a Lower Effort differentiated service that would be lower than best effort [RFC3662]. Such a service could carry traffic for which delivery is strictly optional, or could carry traffic that is important but that has low priority in terms of time. Because it does not interfere with best-effort traffic, Lower Effort services could be used by transport protocols that start up faster than slow-start. For example, [SGF05] is a proposal for the transport sender to use low-priority traffic for much of the initial traffic, with routers configured to use strict priority queueing.

A separate but related issue is that of below-best-effort TCP, variants of TCP that would not rely on Lower Effort services in the network, but would approximate below-best-effort traffic by detecting and responding to congestion sooner than standard TCP. TCP Nice [V02] and TCP Low Priority (TCP-LP) [KK03] are two such proposals for below-best-effort TCP, with the purpose of allowing TCP connections to use the bandwidth unused by TCP and other traffic in a non-intrusive fashion. Both TCP Nice and TCP Low Priority use the default slow-start mechanisms of TCP.

We note that Quick-Start is quite different from either a Lower-Effort service or a below-best-effort variant of TCP. Unlike these proposals, Quick-Start is intended to be useful for best-effort traffic that wishes to receive at least as much bandwidth as competing best-effort connections.

Appendix B. Design Decisions

B.1. Alternate Mechanisms for the Quick-Start Request: ICMP and RSVP

This document has proposed using an IP Option for the Quick-Start Request from the sender to the receiver, and using transport mechanisms for the Quick-Start Response from the receiver back to the sender. In this section, we discuss alternate mechanisms, and consider whether ICMP ([RFC792], [RFC4443]) or RSVP [RFC2205] protocols could be used for delivering the Quick-Start Request.

B.1.1. ICMP

Being a control protocol used between Internet nodes, one could argue that ICMP is the ideal method for requesting permission for faster start-up from routers. The ICMP header is above the IP header. Quick-Start could be accomplished with ICMP as follows: If the ICMP protocol is used to implement Quick-Start, the equivalent of the Quick-Start IP option would be carried in the ICMP header of the ICMP Quick-Start Request. The ICMP Quick-Start Request would have to pass by the routers on the path to the receiver, possibly using the IP Router Alert option [RFC2113]. A router that approves the Quick-Start Request would take the same actions as in the case with the Quick-Start IP Option, and forward the packet to the next router along the path. A router that does not approve the Quick-Start Request, even with a decreased value for the Requested Rate, would delete the ICMP Quick-Start Request, and send an ICMP Reply to the sender that the request was not approved. If the ICMP Reply was dropped in the network, and did not reach the receiver, the sender would still know that the request was not approved from the absence of feedback from the receiver. If the ICMP Quick-Start Request was dropped in the network due to congestion, the sender would assume that the request was not approved. The ICMP message would need the source and destination port numbers for demultiplexing at the end nodes. If the ICMP Quick-Start Request reached the receiver, the receiver would use transport-level or application-level mechanisms to send a response to the sender, exactly as with the IP Option.

One benefit of using ICMP would be that the delivery of the TCP SYN packet or other initial packet would not be delayed by IP option processing at routers. A greater advantage is that if middleboxes were blocking packets with Quick-Start Requests, using the Quick-Start Request in a separate ICMP packet would mean that the middlebox behavior would not affect the connection as a whole. (To get this robustness to middleboxes with TCP using an IP Quick-Start Option, one would have to have a TCP-level Quick-Start Request packet that could be sent concurrently with, but separately from, the TCP SYN packet.)

However, there are a number of disadvantages to using ICMP. Some firewalls and middleboxes may not forward the ICMP Quick-Start Request packets. (If an ICMP Reply packet from a router to the sender is dropped in the network, the sender would still know that the request was not approved, as stated earlier, so this would not be as serious of a problem.) In addition, it would be difficult, if not impossible, for a router in the middle of an IP tunnel to deliver an ICMP Reply packet to the actual source, for example, when the inner IP header is encrypted, as in IPsec ESP tunnel mode [RFC4301]. Again, however, the ICMP Reply packet would not be essential to the correct operation of ICMP Quick-Start.

Unauthenticated out-of-band ICMP messages could enable some types of attacks by third-party malicious hosts that are not possible when the control information is carried in-band with the IP packets that can only be altered by the routers on the connection path. Finally, as a minor concern, using ICMP would cause a small amount of additional traffic in the network, which is not the case when using IP options.

B.1.2. RSVP

With some modifications, RSVP [RFC2205] could be used as a bearer protocol for carrying the Quick-Start Requests. Because routers are expected to process RSVP packets more extensively than the normal transport protocol IP packets, delivering a Quick-Start rate request using an RSVP packet would seem an appealing choice. However, Quick-Start with RSVP would require a few differences from the conventional usage of RSVP. Quick-Start would not require periodical refreshing of soft state, because Quick-Start does not require per-connection state in routers. Quick-Start Requests would be transmitted downstream from the sender to receiver in the RSVP Path messages, which is different from the conventional RSVP model where the reservations originate from the receiver. Furthermore, the Quick-Start Response would be sent using the transport-level or application-level mechanisms, instead of using the RSVP Resv message.

If RSVP was used for carrying a Quick-Start Request, a new "Quick-Start Request" class object would be included in the RSVP Path message that is sent from the sender to receiver. The object would contain the rate request field in addition to the common length and type fields. The Send_TTL field in the RSVP common header could be used as the equivalent of the QS TTL field. The Quick-Start capable routers along the path would inspect the Quick-Start Request object in the RSVP Path message, decrement Send_TTL, and adjust the rate request field if needed. If an RSVP router did not understand the Quick-Start Request object, it would reject the entire RSVP message and send an RSVP PathErr message back to the sender. When an RSVP message with the Quick-Start Request object reaches the receiver, the receiver sends a Quick-Start Reply message in the corresponding transport protocol header in the same way as described in the context of IP options earlier. If the RSVP message with the Quick-Start Request object was dropped along the path, the transport sender would simply proceed with the normal congestion control procedures.

Much of the discussion about benefits and drawbacks of using ICMP for making the Quick-Start Request also applies to the RSVP case. If the Quick-Start Request was transmitted in a separate packet instead of as an IP option, the transport protocol packet delivery would not be delayed due to IP option processing at the routers, and the initial transport packets would reach their destination more reliably. The possible disadvantages of using ICMP and RSVP are also expected to be similar: middleboxes in the network may not be able to forward the Quick-Start Request messages, and the IP tunnels might cause problems for processing the Quick-Start Requests.

B.2. Alternate Encoding Functions

In this section, we look at alternate encoding functions for the Rate Request field in the Quick-Start Request. The main requirements for this function is that it should have a sufficiently wide range for the requested rate. There is no need for overly fine-grained precision in the requested rate. Similarly, while it would be attractive for the encoding function to be easily computable, it is also possible for end-nodes and routers to simply store the table giving the mapping between the value N in the Rate Request field, and the actual rate request f(N). In this section, we consider possible encoding methods for Rate Request fields of different sizes, including four-bit, eight-bit, and larger Rate Request fields.

Linear functions:
One possible proposal would be for the Rate Request field to be formatted in bits per second, scaled so that one unit equals M Kbps, for some fixed value of M. Thus, for the value N in the Rate Request field, the requested rate would be M*N Kbps.

Powers of two:
If a granularity of factors of two is sufficient for the Rate Request, then the encoding function with the most range would be for the requested rate to be K*2^N; for N, the value in the Rate Request field; and for K, some constant. For N=0, the rate request would be set to zero, regardless of the encoding function. For example, for K=40,000 and an eight-bit Rate Request field, the request range would be from 80 Kbps to 40*2^255 Kbps. This clearly would be an unnecessarily large request range.

For a four-bit Rate Request field, the upper limit on the rate request is 1.3 Gbps. It seems to us that an upper limit of 1.3 Gbps would be fine for the Quick-Start rate request, and that connections wishing to start up with a higher initial sending rate should be encouraged to use other mechanisms, such as the explicit reservation of bandwidth. If an upper limit of 1.3 Gbps was not acceptable, then five or six bits could be used for the Rate Request field.

The lower limit of 80 Kbps could be useful for flows with round-trip times of a second or more. For a flow with a round-trip time of one second, as is typical in some wireless networks, the TCP initial window of 4380 bytes allowed by [RFC3390] (given appropriate packet sizes) would translate to an initial sending rate of 35 Kbps. Thus, for TCP flows, a rate request of 80 Kbps could be useful for some flows with large round-trip times.

The lower limit of 80 Kbps could also be useful for some non-TCP flows that send small packets, with at most one small packet every 10 ms. A rate request of 80 Kbps would translate to a rate of a hundred 100-byte packets per second (including packet headers). While some small-packet flows with large round-trip times might find a smaller rate request of 40 Kbps to be useful, our assumption is that a lower limit of 80 Kbps on the rate request will be generally sufficient. Again, if the lower limit of 80 kbps was not acceptable, then extra bits could be used for the Rate Request field.

If the granularity of factors of two was too coarse, then the encoding function could use a base less than two. An alternate form for the encoding function would be to use a hybrid of linear and exponential functions.

A mantissa and exponent representation:
Section 4.4 of [B05] suggests a mantissa and exponent representation for the Quick-Start encoding function. With e and f as the binary numbers in the exponent and mantissa fields, and with 0 <= f < 1, this would represent the rate (1+f)*2^e. [B05] suggests a mantissa field for f of 8, 16, or 24 bits, with an exponent field for e of 8 bits. This representation would allow larger rate requests, with an encoding that is less coarse than the powers-of-two encoding used in this document.

Constraints of the transport protocol:
We note that the Rate Request is also constrained by the abilities of the transport protocol. For example, for TCP with Window Scaling, the maximum window is at most 2**30 bytes. For a TCP connection with a long, 1 second round-trip time, this would give a maximum sending rate of 1.07 Gbps.

B.3. The Quick-Start Request: Packets or Bytes?

One of the design questions is whether the Rate Request field should be in bytes per second or in packets per second. We discuss this separately from the perspective of the transport, and from the perspective of the router.

For TCP, the results from the Quick-Start Request are translated into a congestion window in bytes, using the measured round-trip time and the MSS. This window applies only to the bytes of data payload, and does not include the bytes in the TCP or IP packet headers. Other transport protocols would conceivably use the Quick-Start Request directly in packets per second, or could translate the Quick-Start Request to a congestion window in packets.

The assumption of this document is that the router only approves the Quick-Start Request when the output link is significantly underutilized. For this, the router could measure the available bandwidth in bytes per second, or could convert between packets and bytes by some mechanism.

If the Quick-Start Request was in bytes per second, and applied only to the data payload, then the router would have to convert from bytes per second of data payload, to bytes per second of packets on the wire. If the Rate Request field was in bytes per second, and the sender ended up using very small packets, this could translate to a significantly larger number in terms of bytes per second on the wire. Therefore, for a Quick-Start Request in bytes per second, it makes most sense for this to include the transport and IP headers as well as the data payload. Of course, this will be, at best, a rough approximation on the part of the sender; the transport-level sender might not know the size of the transport and IP headers in bytes, and might know nothing at all about the separate headers added in IP tunnels downstream. This rough estimate seems sufficient, however, given the overall lack of fine precision in Quick-Start functionality.

It has been suggested that the router could possibly use information from the MSS option in the TCP packet header of the SYN packet to convert the Quick-Start Request from packets per second to bytes per second, or vice versa. This would be problematic for several reasons. First, if IPsec is used, the TCP header will be encrypted. Second, the MSS option is defined as the maximum MSS that the TCP sender expects to receive, not the maximum MSS that the TCP sender plans to send [RFC793]. However, it is probably often the case that this MSS also applies as an upper bound on the MSS used by the TCP sender in sending.

We note that the sender does not necessarily know the Path MTU when the Quick-Start Request is sent, or when the initial window of data is sent. Thus, with IPv4, packets from the initial window could end up being fragmented in the network if the "Don't Fragment" (DF) bit is not set [RFC1191]. A Rate Request in bytes per second is reasonably robust to fragmentation. Clearly, a Rate Request in packets per second is less robust in the presence of fragmentation. Interactions between larger initial windows and Path MTU Discovery are discussed in more detail in RFC 3390 [RFC3390].

For a Quick-Start Request in bytes per second, the transport senders would have the additional complication of estimating the bandwidth usage added by the packet headers.

We have chosen a Rate Request field in bytes per second rather than in packets per second because it seems somewhat more robust, particularly to routers.

B.4. Quick-Start Semantics: Total Rate or Additional Rate?

For a Quick-Start Request sent in the middle of a connection, there are two possible semantics for the Rate Request field, as follows:

(1) Total Rate: The requested Rate Request is the requested total rate for the connection, including the current rate; or

(2) Additional Rate: The requested Rate Request is the requested increase in the total rate for that connection, over and above the current sending rate.

When the Quick-Start Request is sent after an idle period, the current sending rate is zero, and there is no difference between (1) and (2) above. However, a Quick-Start Request can also be sent in the middle of a connection that has not been idle, e.g., after a mobility event, or after an application-limited period when the sender is suddenly ready to send at a much higher rate. In this case, there can be a significant difference between (1) and (2) above. In this section, we consider briefly the tradeoffs between these two options, and explain why we have chosen the 'Total Rate' semantics.

The Total Rate semantics makes it easier for routers to "allocate" the same rate to all connections. This lends itself to fairness, and improves convergence times between old and new connections. With the Additional Rate semantics, the router would not necessarily know the current sending rates of the flows requesting additional rates, and therefore would not have sufficient information to use fairness as a metric in granting rate requests. With the Total Rate semantics, the fairness is automatic; the router is not granting rate requests for *additional* bandwidth without knowing the current sending rates of the different flows.

The Additional Rate semantics also lends itself to gaming by the connection, with senders sending frequent Quick-Start Requests in the hope of gaining a higher rate. If the router is granting the same maximum rate for all rate requests, then there is little benefit to a connection of sending a rate request over and over again. However, if the router is granting an *additional* rate with each rate request, over and above the current sending rate, then it is in a connection's interest to send as many rate requests as possible, even if very few of them are, in fact, granted.

Appendix E discusses a Report of Current Sending Rate as one possible function in the Quick-Start Option. However, we have not standardized this possible use at this time.

B.5. Alternate Responses to the Loss of a Quick-Start Packet

Section 4.6 discusses TCP's response to the loss of a Quick-Start packet in the initial window. This section discusses several alternate responses.

One possible alternative to reverting to the default Slow-Start after the loss of a Quick-Start packet from the initial window would have been to halve the congestion window and continue in congestion avoidance. However, we note that this would not have been a desirable response for either the connection or for the network as a whole. The packet loss in the initial window indicates that Quick-Start failed in finding an appropriate congestion window, meaning that the congestion window after halving may easily also be wrong.

A more moderate alternate would be to continue in congestion avoidance from a window of (W-D)/2, where W is the Quick-Start congestion window, and D is the number of packets dropped or marked from that window. However, such an approach would implicitly assume that the number of Quick-Start packets delivered is a good indication of the appropriate available bandwidth for that flow, even though other packets from that window were dropped in the network. And, further, that using half the number of segments that were successfully transmitted is conservative enough to account for the possibly inaccurate congestion window indication. We believe that such an assumption would require more analysis at this point, particularly in a network with a range of packet dropping mechanisms at the router, and we cannot recommend it at this time.

Another drawback of approaches that don't revert back to slow-start when a Quick-Start packet in the initial window is dropped is that such approaches could give the TCP receiver a greater incentive to lie about the Quick-Start Request. If the sender reverts to slow-start when a Quick-Start packet in the initial window is dropped, this diminishes the benefit a receiver would get from a Quick-Start request that resulted in a dropped or ECN-marked packet.

B.6. Why Not Include More Functionality?

This proposal for Quick-Start is a rather coarse-grained mechanism that would allow a connection to use a higher sending rate along underutilized paths, but that does not attempt to provide a next-generation transport protocol or congestion control mechanism, and does not attempt the goal of providing very high throughput with very low delay. Appendix A.4 discusses a number of proposals (such as XCP, MaxNet, and AntiECN) that provide more fine-grained per-packet feedback from routers than the current congestion control mechanisms and that attempt these more ambitious goals.

Compared to proposals such as XCP and AntiECN, Quick-Start offers much less control. Quick-Start does not attempt to provide a new congestion control mechanism, but simply to get permission from routers for a higher sending rate at start-up, or after an idle period. Quick-Start can be thought of as an "anti-congestion-control" mechanism that is only of any use when all the routers along the path are significantly underutilized. Thus, Quick-Start is of no use towards a target of high link utilization, or fairness in a high-utilization scenario, or controlling queueing delay during high utilization, or anything of the like.

At the same time, Quick-Start would allow larger initial windows than would proposals such as AntiECN, requires less input to routers than XCP (e.g., XCP's cwnd and RTT fields), and would require less frequent feedback from routers than any new congestion control mechanism. Thus, Quick-Start is significantly less powerful than proposals for new congestion control mechanisms, such as XCP and AntiECN, but as powerful or more powerful in terms of the specific issue of allowing larger initial windows. Also, (we think) it is more amenable to incremental deployment in the current Internet.

We do not discuss proposals such as XCP in detail, but simply note that there are a number of open questions. One question concerns whether there is a pressing need for more sophisticated congestion control mechanisms, such as XCP, in the Internet. Quick-Start is inherently a rather crude tool that does not deliver assurances about maintaining high link utilization and low queueing delay; Quick-Start is designed for use in environments that are significantly underutilized, and addresses the single question of whether a higher sending rate is allowed. New congestion control mechanisms with more fine-grained feedback from routers could allow faster start-ups even in environments with rather high link utilization. Is this a pressing requirement? Are the other benefits of more fine-grained congestion control feedback from routers a pressing requirement?

We would argue that even if more fine-grained per-packet feedback from routers was implemented, it is reasonable to have a separate mechanism, such as Quick-Start, for indicating an allowed initial sending rate, or an allowed total sending rate after an idle or underutilized period.

One difference between Quick-Start and current proposals for fine-grained per-packet feedback, such as XCP, is that XCP is designed to give robust performance even in the case where different packets within a connection routinely follow different paths. XCP achieves relatively robust performance in the presence of multipath routing by using per-packet feedback, where the feedback carried in a single packet is about the relative increase or decrease in the rate or window to take effect when that particular packet is acknowledged, not about the allowed sending rate for the connection as a whole.

In contrast, Quick-Start sends a single Quick-Start Request, and the answer to that request gives the allowed sending rate for an entire window of data. As a result, Quick-Start could be problematic in an environment where some fraction of the packets in a window of data take path A, and the rest of the packets take path B; for example, the Quick-Start Request could have traveled on path A, while half the Quick-Start packets sent in the succeeding round-trip time are routed on path B. We note that [ZDPS01] shows Internet paths to be stable on the order of RTTs.

There are also differences between Quick-Start and some of the proposals for per-packet feedback in terms of the number of bits of feedback required from the routers to the end-nodes. Quick-Start uses four bits of feedback in the rate request field to indicate the allowed sending rate. XCP allocates a byte for per-packet feedback, though there has been discussion of variants of XCP with less per-packet feedback. This would be more like other proposals, such as anti-ECN, that use a single bit of feedback from routers to indicate that the sender can increase as fast as slow-starting, in response to this particular packet acknowledgement. In general, there is probably considerable power in fine-grained proposals with only two bits of feedback, indicating that the sender should decrease, maintain, or increase the sending rate or window when this packet is acknowledged. However, the power of Quick-Start would be considerably limited if it was restricted to only two bits of feedback; it seems likely that determining the initial sending rate fundamentally requires more bits of feedback from routers than does the steady-state, per-packet feedback to increase or decrease the sending rate.

On a more practical level, one difference between Quick-Start and proposals for per-packet feedback is that there are fewer open issues with Quick-Start than there would be with a new congestion control mechanism. Because Quick-Start is a mechanism for requesting an initial sending rate in an underutilized environment, its fairness issues are less severe than those of a general congestion control mechanism. With Quick-Start, there is no need for the end nodes to tell the routers the round-trip time and congestion window, as is done in XCP; all that is needed is for the end nodes to report the requested sending rate.

Table 3 provides a summary of the differences between Quick-Start and proposals for per-packet congestion control feedback.

	Quick-Start	Proposals for Per-Packet Feedback
Semantics:	Allowed sending rate per connection.	Change in rate/window, per-packet.
Relationship to congestion ctrl:	In addition.	Replacement.
Frequency:	Start-up, or after an idle period.	Every packet.
Limitations:	Only useful on underutilized paths.	General congestion control mechanism.
Input to routers:	Rate request.	RTT, cwnd, request (XCP) None (Anti-ECN).
Bits of feedback:	Four bits for rate request.	A few bits would suffice?

Table 3: Differences between Quick-Start and Proposals for Fine-Grained Per-Packet Feedback.

A separate question concerns whether mechanisms, such as Quick-Start, in combination with HighSpeed TCP and other changes in progress, would make a significant contribution towards meeting some of these needs for new congestion control mechanisms. This could be viewed as a positive step towards meeting some of the more pressing current needs with a simple and reasonably deployable mechanism, or alternately, as a negative step of unnecessarily delaying more fundamental changes. Without answering this question, we would note that our own approach tends to favor the incremental deployment of relatively simple mechanisms, as long as the simple mechanisms are not short-term hacks, but mechanisms that lead the overall architecture in the fundamentally correct direction.

B.7. Alternate Implementations for a Quick-Start Nonce

B.7.1. An Alternate Proposal for the Quick-Start Nonce

An alternate proposal for the Quick-Start Nonce from [B05] would be for an n-bit field for the QS Nonce, with the sender generating a random nonce when it generates a Quick-Start Request. Each router that reduces the Rate Request by r would hash the QS nonce r times, using a one-way hash function such as MD5 [RFC1321] or the secure hash 1 [SHA1]. The receiver returns the QS nonce to the sender. Because the sender knows the original value for the nonce, and the original rate request, the sender knows the total number of steps s that the rate has been reduced. The sender then hashes the original nonce s times to check whether the result is the same as the nonce returned by the receiver.

This alternate proposal for the nonce would be considerably more powerful than the QS nonce described in Section 3.4, but it would also require more CPU cycles from the routers when they reduce a Quick-Start Request, and from the sender in verifying the nonce returned by the receiver. As reported in [B05], routers could protect themselves from processor exhaustion attacks by limiting the rate at which they will approve reductions of Quick-Start Requests.

Both the Function field and the Reserved field in the Quick-Start Option would allow the extension of Quick-Start to use Quick-Start requests with the alternate proposal for the Quick-Start Nonce, if it was ever desired.

B.7.2. The Earlier Request-Approved Quick-Start Nonce

An earlier version of this document included a Request-Approved Quick-Start Nonce (QS Nonce) that was initialized by the sender to a non-zero, 'random' eight-bit number, along with a QS TTL that was initialized to the same value as the TTL in the IP header. The Request-Approved Quick-Start Nonce would have been returned by the transport receiver to the transport sender in the Quick-Start Response. A router could deny the Quick-Start Request by failing to decrement the QS TTL field, by zeroing the QS Nonce field, or by deleting the Quick-Start Request from the packet header. The QS Nonce was included to provide some protection against broken downstream routers, or against misbehaving TCP receivers that might be inclined to lie about whether the Rate Request was approved. This protection is now provided by the QS Nonce, by the use of a random initial value for the QS TTL field, and by Quick-Start-capable routers hopefully either deleting the Quick-Start Option or zeroing the QS TTL and QS Nonce fields when they deny a request.

With the old Request-Approved Quick-Start Nonce, along with the QS TTL field set to the same value as the TTL field in the IP header, the Quick-Start Request mechanism would have been self-terminating; the Quick-Start Request would terminate at the first participating router after a non-participating router had been encountered on the path. This minimizes unnecessary overhead incurred by routers because of option processing for the Quick-Start Request. In the current specification, this "self-terminating" property is provided by Quick-Start-capable routers hopefully either deleting the Quick-Start Option or zeroing the Rate Request field when they deny a request. Because the current specification uses a random initial value for the QS TTL, Quick-Start-capable routers can't tell if the Quick-Start Request is invalid because of non-Quick-Start-capable routers upstream. This is the cost of using a design that makes it difficult for the receiver to cheat about the value of the QS TTL.

Appendix C. Quick-Start with DCCP

DCCP is a new transport protocol for congestion-controlled, unreliable datagrams, intended for applications such as streaming media, Internet telephony, and online games. In DCCP, the application has a choice of congestion control mechanisms, with the currently-specified Congestion Control Identifiers (CCIDs) being CCID 2 for TCP-like congestion control, and CCID 3 for TCP Friendly Rate Control (TFRC), an equation-based form of congestion control. We refer the reader to [RFC4340] for a more detailed description of DCCP and congestion control mechanisms.

Because CCID 3 uses a rate-based congestion control mechanism, it raises some new issues about the use of Quick-Start with transport protocols. In this document, we don't attempt to specify the use of Quick-Start with DCCP. However, we do discuss some of the issues that might arise.

In considering the use of Quick-Start with CCID 3 for requesting a higher initial sending rate, the following questions arise:

(1) How does the sender respond if a Quick-Start packet is dropped?

As in TCP, if an initial Quick-Start packet is dropped, the CCID 3 sender should revert to the congestion control mechanisms it would have used if the Quick-Start Request had not been approved.

(2) When does the sender decide there has been no feedback from the receiver?

Unlike TCP, CCID 3 does not use acknowledgements for every packet, or for every other packet. In contrast, the CCID 3 receiver sends feedback to the sender roughly once per round-trip time. In CCID 3, the allowed sending rate is halved if no feedback is received from the receiver in at least four round-trip times (when the sender is sending at least one packet every two round-trip times). When a Quick-Start Request is used, it would seem necessary to use a smaller time interval, e.g., to reduce the sending rate if no feedback arrives from the receiver in at least two round-trip times.

The question also arises of how the sending rate should be reduced after a period of no feedback from the receiver. As with TCP, the default CCID 3 response of halving the sending rate is not necessarily a sufficient response to the absence of feedback; an alternative is to reduce the sending rate to the sending rate that would have been used if no Quick-Start Request had been approved. That is, if a CCID 3 sender uses a Quick-Start Request, special rules might be required to handle the sender's response to a period of no feedback from the receiver regarding the Quick-Start packets.

Similarly, in considering the use of Quick-Start with CCID 3 for requesting a higher sending rate after an idle period, the following questions arise:

(1) What rate does the sender request?

As in TCP, there is a straightforward answer to the rate request that the CCID 3 sender should use in requesting a higher sending rate after an idle period. The sender knows the current loss event rate, either from its own calculations or from feedback from the receiver, and can determine the sending rate allowed by that loss event rate. This is the upper bound on the sending rate that should be requested by the CCID 3 sender. A Quick-Start Request is useful with CCID 3 when the sender is coming out of an idle or underutilized period, because in standard operation, CCID 3 does not allow the sender to send more than twice as fast as the receiver has reported received in the most recent feedback message.

(2) What is the response to loss?

The response to the loss of Quick-Start packets should be to return to the sending rate that would have been used if Quick-Start had not been requested.

(3) When does the sender decide there has been no feedback from the receiver?

As in the case of the initial sending rate, it would seem prudent to reduce the sending rate if no feedback is received from the receiver in at least two round-trip times. It seems likely that, in this case, the sending rate should be reduced to the sending rate that would have been used if no Quick-Start Request had been approved.

Appendix D. Possible Router Algorithm

This specification does not tightly define the algorithm a router uses when deciding whether to approve a Quick-Start Rate Request or whether and how to reduce a Rate Request. A range of algorithms is likely useful in this space and we consider the algorithm a particular router uses to be a local policy decision. In addition, we believe that additional experimentation with router algorithms is necessary to have a solid understanding of the dynamics various algorithms impose. However, we provide one particular algorithm in this appendix as an example and as a framework for thinking about additional mechanisms.

[SAF06] provides several algorithms routers can use to consider incoming Rate Requests. The decision process involves two algorithms. First, the router needs to track the link utilization over the recent past. Second, this utilization needs to be updated by the potential new bandwidth from recent Quick-Start approvals, and then compared with the router's notion of when it is underutilized enough to approve Quick-Start Requests (of some size).

First, we define the "peak utilization" estimation technique (from [SAF06]). This mechanism records the utilization of the link every S seconds and stores the most recent N of these measurements. The utilization is then taken as the highest utilization of the N samples. This method, therefore, keeps N*S seconds of history. This algorithm reacts rapidly to increases in the link utilization. In [SAF06], S is set to 0.15 seconds, and experiments use values for N ranging from 3 to 20.

Second, we define the "target" algorithm for processing incoming Quick-Start Rate Requests (also from [SAF06]). The algorithm relies on knowing the bandwidth of the outgoing link (which, in many cases, can be easily configured), the utilization of the outgoing link (from an estimation technique such as given above), and an estimate of the potential bandwidth from recent Quick-Start approvals.

Tracking the potential bandwidth from recent Quick-Start approvals is another case where local policy dictates how it should be done. The simplest method, outlined in Section 8.2, is for the router to keep track of the aggregate Quick-Start rate requests approved in the most recent two or more time intervals, including the current time interval, and to use the sum of the aggregate rate requests over these time intervals as the estimate of the approved Rate Requests. The experiments in [SAF06] keep track of the aggregate approved Rate Requests over the most recent two time intervals, for intervals of 150 msec.

The target algorithm also depends on a threshold (qs_thresh) that is the fraction of the outgoing link bandwidth that represents the router's notion of "significantly underutilized". If the utilization, augmented by the potential bandwidth from recent Quick-Start approvals, is above this threshold, then no Quick-Start Rate Requests will be approved. If the utilization, again augmented by the potential bandwidth from recent Quick-Start approvals, is less than the threshold, then Rate Requests can be approved. The Rate Requests will be reduced such that the bandwidth allocated would not drive the utilization to more than the given threshold. The algorithm is:

```
util_bw = bandwidth * utilization;
util_bw = util_bw + recent_qs_approvals;
if (util_bw < (qs_thresh * bandwidth))
{
    approved = (qs_thresh * bandwidth) - util_bw;
    if (rate_request < approved)
        approved = rate_request;
    approved = round_down (approved);
    recent_qs_approvals += approved;
}
```

Also note that, given that Rate Requests are fairly coarse, the approved rate should be rounded down when it does not fall exactly on one of the rates allowed by the encoding scheme.

Routers that wish to keep close track of the allocated Quick-Start bandwidth could use Approved Rate reports to learn when rate requests had been decremented downstream in the network, and also to learn when a sender begins to use the approved Quick-Start Request.

Appendix E. Possible Additional Uses for the Quick-Start Option

The Quick-Start Option contains a four-bit Function field in the third byte, enabling additional uses to be defined for the Quick-Start Option. In this section, we discuss some of the possible additional uses that have been discussed for Quick-Start. The Function field makes it easy to add new functions for the Quick-Start Option.

Report of Current Sending Rate: A Quick-Start Request with the 'Report of Current Sending Rate' codepoint set in the Function field would be using the Rate Request field to report the current estimated sending rate for that connection. This could accompany a second Quick-Start Request in the same packet containing a standard rate request, for a connection that is using Quick-Start to increase its current sending rate.

Request to Increase Sending Rate: A codepoint for 'Request to Increase Sending Rate' in the Function field would indicate that the connection is not idle or just starting up, but is attempting to use Quick-Start to increase its current sending rate. This codepoint would not change the semantics of the Rate Request field.

RTT Estimate: If a codepoint for 'RTT Estimate' was used, a field for the RTT Estimate would contain one or more bits giving the sender's rough estimate of the round-trip time, if known. E.g., the sender could estimate whether the RTT was greater or less than 200 ms. Alternately, if the sender had an estimate of the RTT when it sends the Rate Request, the two-bit Reserved field at the end of the Quick-Start Option could be used for a coarse-grained encoding of the RTT.

Informational Request: An Informational Request codepoint in the Function field would indicate that a request is purely informational, for the sender to find out if a rate request would be approved, and what size rate request would be approved when the Informational Request is sent. For example, an Informational Request could be followed one round-trip time later by a standard Quick-Start Request. A router approving an Informational Request would not consider this as an approval for Quick-Start bandwidth to be used, and would not be under any obligation to approve a similar standard Quick-Start Request one round-trip time later. An Informational Request with a rate request of zero could be used by the sender to find out if all of the routers along the path supported Quick-Start.

Use Format X for the Rate Request Field: A Quick-Start codepoint for 'Use Format X for the Rate Request Field' would indicate that an alternate format was being used for the Rate Request field.

Do Not Decrement: A Do Not Decrement codepoint could be used for a Quick-Start Request where the sender would rather have the request denied than to have the rate request decremented in the network. This could be used if the sender was only interested in using Quick-Start if the original rate request was approved.

Temporary Bandwidth Use: A Temporary codepoint has been proposed to indicate that a connection would only use the requested bandwidth for a single time interval.

Normative References

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 3390, October 2002.

[RFC3742] Floyd, S., "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, March 2004.

Informative References

[RFC792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, April 1997.

[RFC2205] Braden, R., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2415] Poduri, K. and K. Nichols, "Simulation Studies of Increased Initial TCP Window Size", RFC 2415, September 1998.

[RFC2488] Allman, M., Glover, D., and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", BCP 28, RFC 2488, January 1999.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol 'L2TP'", RFC 2661, August 1999.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

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

[RFC3360] Floyd, S., "Inappropriate TCP Resets Considered Harmful", BCP 60, RFC 3360, August 2002.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, December 2003.

[RFC3697] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC3819] Karn, P., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, July 2004.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[AHO98] M. Allman, C. Hayes and S. Ostermann. An evaluation of TCP with Larger Initial Windows. ACM Computer Communication Review, July 1998.

[B05] Briscoe, B., "Review: Quick-Start for TCP and IP", <http://www.cs.ucl.ac.uk/staff/B.Briscoe/pubs.html>, November 2005.

[BW97] G. Brasche and B. Walke. Concepts, Services and Protocols of the new GSM Phase 2+ General Packet Radio Service. IEEE Communications Magazine, pages 94--104, August 1997.

[GPAR02] A. Gurtov, M. Passoja, O. Aalto, and M. Raitola. Multi-Layer Protocol Tracing in a GPRS Network. In Proceedings of the IEEE Vehicular Technology Conference (Fall VTC2002), Vancouver, Canada, September 2002.

[H05] P. Hoffman, email, October 2005. Citation for acknowledgement purposes only.

[HKP01] M. Handley, C. Kreibich and V. Paxson, Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, Proc. USENIX Security Symposium 2001.

[Jac88] V. Jacobson, Congestion Avoidance and Control, ACM SIGCOMM.

[JD02] Manish Jain, Constantinos Dovrolis, End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput, SIGCOMM 2002.

[K03] S. Kunniyur, "AntiECN Marking: A Marking Scheme for High Bandwidth Delay Connections", Proceedings, IEEE ICC '03, May 2003. <http://www.seas.upenn.edu/~kunniyur/>.

[KAPS02] Rajesh Krishnan, Mark Allman, Craig Partridge, James P.G. Sterbenz. Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks. Technical Report No. 8333, BBN Technologies, March 2002. <http://www.icir.org/mallman/papers/>.

[KHR02] Dina Katabi, Mark Handley, and Charles Rohrs, Internet Congestion Control for Future High Bandwidth-Delay Product Environments. ACM Sigcomm 2002, August 2002. <http://ana.lcs.mit.edu/dina/XCP/>.

[KK03] A. Kuzmanovic and E. W. Knightly. TCP-LP: A Distributed Algorithm for Low Priority Data Transfer. INFOCOM 2003, April 2003.

[KSEPA04] Rajesh Krishnan, James Sterbenz, Wesley Eddy, Craig Partridge, Mark Allman. Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks. Computer Networks, 46(3), October 2004.

[L05] Guohan Lu, Nonce in TCP Quick Start, September 2005. <http://www.net-glyph.org/~lgh/nonce-usage.pdf>.

[MH06] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", Work in Progress, December 2006.

[MAF04] Alberto Medina, Mark Allman, and Sally Floyd, Measuring Interactions Between Transport Protocols and Middleboxes, Internet Measurement Conference 2004, August 2004. <http://www.icir.org/tbit/".

[MAF05] Alberto Medina, Mark Allman, and Sally Floyd. Measuring the Evolution of Transport Protocols in the Internet. Computer Communications Review, April 2005.

[MaxNet] MaxNet Home Page, <http://netlab.caltech.edu/~bartek/maxnet.htm>.

[P00] Joon-Sang Park, Bandwidth Discovery of a TCP Connection, report to John Heidemann, 2000, private communication. Citation for acknowledgement purposes only.

[PABL+05] Ruoming Pang, Mark Allman, Mike Bennett, Jason Lee, Vern Paxson, Brian Tierney. A First Look at Modern Enterprise Traffic. ACM SIGCOMM/USENIX Internet Measurement Conference, October 2005.

[PRAKS02] Craig Partridge, Dennis Rockwell, Mark Allman, Rajesh Krishnan, James P.G. Sterbenz. A Swifter Start for TCP. Technical Report No. 8339, BBN Technologies, March 2002. <http://www.icir.org/mallman/papers/>.

[RW03] Mattia Rossi and Michael Welzl, On the Impact of IP Option Processing, Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 15, Institute of Computer Science, University of Innsbruck, Austria, October 2003.

[RW04] Mattia Rossi and Michael Welzl, On the Impact of IP Option Processing - Part 2, Preprint-Reihe des Fachbereichs Mathematik - Informatik, No. 26, Institute of Computer Science, University of Innsbruck, Austria, July 2004.

[S02] Ion Stoica, private communication, 2002. Citation for acknowledgement purposes only.

[SAF06] Pasi Sarolahti, Mark Allman, and Sally Floyd. Determining an Appropriate Sending Rate Over an Underutilized Network Path. February 2006. <http://www.icir.org/floyd/quickstart.html>.

[SGF05] Singh, M., Guha, S., and P. Francis, "Utilizing spare network bandwidth to improve TCP performance", ACM SIGCOMM 2005 Work in Progress session, August 2005, <https://www.guha.cc/saikat/pub/sigcomm05-lowtcp.pdf>.

[SHA1] "Secure Hash Standard", FIPS, U.S. Department of Commerce, Washington, D.C., publication 180-1, April 1995.

[SH02] Srikanth Sundarrajan and John Heidemann. Study of TCP Quick Start with NS-2. Class Project, December 2002. Not publicly available; citation for acknowledgement purposes only.

[V02] A. Venkataramani, R. Kokku, and M. Dahlin. TCP Nice: A Mechanism for Background Transfers. OSDI 2002.

[VH97] V. Visweswaraiah and J. Heidemann, Improving Restart of Idle TCP Connections, Technical Report 97-661, University of Southern California, November 1997.

[W00] Michael Welzl: PTP: Better Feedback for Adaptive Distributed Multimedia Applications on the Internet, IPCCC 2000 (19th IEEE International Performance, Computing, And Communications Conference), Phoenix, Arizona, USA, 20-22 February 2000. <http://www.welzl.at/research/publications/>.

[ZDPS01] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, On the Constancy of Internet Path Properties, Proc. ACM SIGCOMM Internet Measurement Workshop, November 2001.

[ZQK00] Y. Zhang, L. Qiu, and S. Keshav, Speeding Up Short Data Transfers: Theory, Architectural Support, and Simulation Results, NOSSDAV 2000, June 2000.

Authors' Addresses

Sally Floyd
Phone: +1 (510) 666-2989
ICIR (ICSI Center for Internet Research)
EMail: floyd@icir.org
URL: http://www.icir.org/floyd/

Mark Allman
ICSI Center for Internet Research
1947 Center Street, Suite 600
Berkeley, CA 94704-1198
Phone: (440) 235-1792
EMail: mallman@icir.org
URL: http://www.icir.org/mallman/

Amit Jain
F5 Networks
EMail: a.jain@f5.com

Pasi Sarolahti
Nokia Research Center
P.O. Box 407
FI-00045 NOKIA GROUP
Finland
Phone: +358 50 4876607
EMail: pasi.sarolahti@iki.fi

Acknowledgement

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