Internet Engineering Task Force — **A. Jain**
INTERNET-DRAFT — **F5 Networks**
draft-ietf-tsvwg-quickstart-00.ps — **S. Floyd**
Expires: November 2005 — **M. Allman**
ICIR
P. Sarolahti
Nokia Research Center
31 May 2005

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. While Quick-Start is designed to be used by a range of transport protocols, in this document we describe its use with TCP. By using Quick-Start, a TCP host, say, host A, would indicate its desired sending rate in bytes per second, using a Quick Start Request 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. If the Quick-Start request is not approved, then the sender would use the default congestion control mechanisms. The Quick-Start mechanism can determine if there are routers along the path that do not understand the Quick-Start Request 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. Quick-Start is designed to allow connections to use higher sending rates when there is significant unused bandwidth along the path, and all of

the routers along the path support the Quick-Start Request.

TO BE DELETED BY THE RFC EDITOR UPON PUBLICATION:

Changes from draft-amit-quick-start-04.txt:
* A significant amount of general editing.
* Because the Rate Request field only uses four bits, specified that the other four bits are reserved, and talked about a possible use for them. This is discussed in a new section on "A Rate-Reduced Nonce?"
* Specified that a Quick-Start-capable router denying a request SHOULD delete the Quick-Start option, and if this is not possible, SHOULD zero the QS TTL and the Rate Request fields.
* Made the following change: If the Quick-Start Response is lost in the network, it is not retransmitted.
* For PMTUD, in Section 4.6, added a suggestion to send one large packet in the initial window for PMTUD, and to send the other packets at 576 bytes.
* Added a paragraph to Section 4.6.3 on retransmitted SYN packets, saying they should use an RTO of three seconds and a new ISN on the retransmitted SYN packet.
* Added that "TCP SHOULD NOT use Quick-Start" after an application-limited period at this time, in Section 4.1, in addition to the old sentence that this "requires further thought and investigation".
* Added an appendix on "Possible Router Algorithm".
* Moved the section on "Quick-Start with DCCP" to the appendix.
* Name changed from draft-amit-quick-start-04.txt to draft-tsvwg-quickstart-00.txt.

Changes from draft-amit-quick-start-03.txt:
* Added a citation to the paper on "Evaluating Quick-Start for TCP", and added pointers to the work in that paper. This work includes:
  - Discussions of router algorithms.
  - Discussions of sizing Quick-Start requests.
* Added sections on "Misbehaving Middleboxes", and on "Attacks on Quick-Start".

Changes from draft-amit-quick-start-02.txt:
* Added a discussion on Using Quick-Start in the Middle of a Connection. The request would be on the total rate, not on the additional rate.
* Changed name "Initial Rate" to "Rate Request", and changed the units from packets per second to bytes per second.
* The following sections are new:
  - The Quick-Start Request Option for IPv6
  - Quick-Start in IP Tunnels
  - When to Use Quick-Start
  - TCP: Responding to a Loss of a Quick-Start Packet
  - TCP: A Quick-Start Request for a Larger Initial Window
  - TCP: A Quick-Start Request after an Idle Period
  - The Quick-Start Mechanisms in DCCP and other Transport

Protocols
- Quick-Start with DCCP
- Implementation and Deployment Issues
- Design Decisions
* Added a discussion of Kunniyur's Anti-ECN proposal.
* Added a section on simulations, with a brief discussion of the simulations by Srikanth Sundarrajan.

Changes from draft-amit-quick-start-01.txt:
* Added a discussion in the related work section about the possibility of optimistically sending a large initial window, without explicit permission of routers.
* Added a discussion in the related work section about the tradeoffs of XCP vs. Quick-Start.
* Added a section on "The Quick-Start Request: Packets or Bytes?"

Changes from draft-amit-quick-start-00.txt:
* The addition of a citation to [KHR02].
* The addition of a Related Work section.
* Deleted the QS Nonce, in favor of a random initial value for the QS TTL.

Table of Contents

1. Introduction

Each TCP connection begins with a question: "What is the appropriate sending rate for the current network path?" The question is not answered explicitly for TCP, but each TCP connection determines the sending rate by probing the network path and altering the congestion window (cwnd) based on perceived congestion. Each connection starts with a pre-configured initial congestion window (ICW). Currently, TCP allows an initial window of between one and four MSS-sized segments [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 appropriate in the current network conditions.

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 of 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.

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.

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 get 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.

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. [ZPS00] shows this assumption should be generally valid.

* Any new mechanism must be incrementally deployable, and might not be supported by all of 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.

General Principles:

* Our underlying premise is that explicit feedback from all of 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 request for a higher sending rate 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.

There are also a number of questions regarding the Quick-Start mechanism that are discussed later in this document.

Open Questions:

* Would the benefits of the Quick-Start mechanism be worth the added complexity?

The benefits and drawbacks of Quick-Start are discussed in more detail in Section 6 on "Evaluation of Quick-Start".

* One practical consideration is that packets with known and unknown IP options are often dropped in the current Internet [MAF04].

This does not preclude using Quick-Start in Intranets. Further, [MAF04] also shows that over time the blocking of packets negotiating ECN has become less common, and therefore an incremental deployment story for Quick-Start based on IP Options is not out of the question. Appendix A.1 on "Alternate Mechanisms for the Quick-Start Request" discusses the possibility of using RSVP or ICMP instead of IP Options for carrying Quick-Start Requests to routers.

* A second practical consideration is that packets could be dropped at non-IP queues along the path.

This is discussed in more detail in Section 6.2. * Apart from the merits and shortcomings of the Quick-Start mechanism, is there likely to be a compelling need to add explicit congestion-related feedback from routers over and above the one-bit feedback from ECN?

If the answer to the question above is yes, should we be considering ways to incorporate Quick-Start in mechanisms that, while more complex, are also sufficiently more powerful than Quick-Start, or should Quick-Start be considered as orthogonal to such mechanisms? This is discussed further in Appendix A.6 on "Why Not Include More Functionality".

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 can 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 Request (QSR) 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. In addition, Quick-Start assumes a unicast, congestion-controlled transport protocol; we do not consider the use of Quick-Start for multicast traffic.

The Quick-Start Request Option includes a request for a sending rate in bytes per second, and a Quick-Start TTL (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 and information about the TTL to the sender using transport-level mechanisms. 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 this information to determine if all of the routers along the path decremented the Quick-Start TTL, approving the Quick-Start Request.

If the request is approved by all of 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 both routers along the path approving the Quick-Start Request. 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 -->
   |
   |              Decrement
   |              QS TTL
   |              to approve
   |              request -->
   |
   |                               Decrement
   |                               QS TTL
   |                               to approve
   |                               request -->
   |
   |                                            <IP TTL: 61>
   |                                            <QS TTL: 89>
   |                                            <TTL Diff: 28>
   |                                            Return Quick-Start
   |                                             info to sender in
   |                                           <-- TCP ACK packet.
   |
   | <TTL Diff: 28>
   | Quick-Start approved,
   | translate to cwnd.
   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 -->
    |
    |                Decrement
    |                QS TTL
    |                to approve
    |                request -->
    |
    |                               Forward packet
    |                               without modifying
    |                               Quick-Start Option. -->
    |
    |                                            <IP TTL: 61>
    |                                            <QS TTL: 90>
    |                                            <TTL Diff: 29>
    |                                            Return Quick-Start
    |                                             info to sender in
    |                                           <-- TCP ACK packet.
    |
    | <TTL Diff: 29>
    | Quick-Start not approved.
    V Use default initial cwnd. -->

            Figure 2: An unsuccessful Quick-Start Request.
```

3. The Quick-Start Request in IP

3.1. The Quick-Start Request Option for IPv4

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

```
        0              1              2              3
+--------------+--------------+--------------+--------------+
| Option       | Length=4     |   QS TTL     |Resv. |Rate   |
|              |              |              |      |Request|
+--------------+--------------+--------------+--------------+

Figure 3.  The Quick-Start Request Option for IPv4.
```

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 (to be assigned by IANA).

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

The third 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). The QS TTL is used by the sender to detect if all of the routers along the path understood and approved the Quick-Start option.

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)

The fourth byte includes a four-bit Reserved field, and a four-bit Rate Request field. 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, 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 A.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 the Rate Request field to the 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.

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.

In IPv4, a change in IP options at routers requires recalculating the IP header checksum.

3.2. The Quick-Start Request Option for IPv6

The Quick-Start Request Option for IPv6 is placed in the Hop-by-Hop Options extension header that is processed at every network node along the communication path [RFC 2460]. The option format following the generic Hop-by-Hop Options header is similar 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 2.

```
        0               1               2               3
   +---------------+---------------+---------------+---------------+
   |   Option      |  Length=2     |   QS TTL      |Resv.  |Rate   |
   |               |               |               |       |Request|
   +---------------+---------------+---------------+---------------+

   Figure 4.  The Quick-Start Request Option for IPv6.
```

The transport receiver compares the Quick-Start TTL with the IPv6 Hop Limit field in order 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 Request 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.

Note that [RFC2460] specifies that when a specific flow label has been assigned to packets, the contents of the Hop-by-Hop options, excluding the next header field, must originate with the same contents throughout the IP flow lifetime. This requirement would have to be modified to implement Quick-Start on an IPv6 implementation that uses flow labels, because the Quick-Start Request option would be included in only a small fraction of the packets during a flow lifetime.

3.3. Processing the Quick-Start Request at Routers

Each participating router can either terminate or approve the Quick-Start Request. The router terminates the Quick-Start Request if the router is not underutilized, and therefore has decided not to grant the Quick-Start Request.

A router that wishes to terminate the Quick-Start Request SHOULD delete the Quick-Start Request from the IP header. This saves resources as downstream routers will have no option to process. If a Quick-Start-capable router wishes to deny the request but doesn't delete the Quick-Start Request from the IP header, then the router SHOULD zero the QS TTL and the Rate Request fields. This may be more efficient for routers to implement than deleting the Quick-Start option. A router that doesn't understand the Quick-Start option will of course simply forward the packet with the Quick-Start Request unchanged.

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

* The router MUST decrements the QS TTL by one.

* If the router is only willing to approve an 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.

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

A non-participating router forwards the Quick-Start Request unchanged, without decrementing the QS TTL. Of course, 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 modifies or deletes the Quick-Start Request in the IPv4 header also MUST update the IPv4 Header checksum. For IPv6, no checksum updates are needed.

3.4. 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. As an example, the router could 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 was to increase by the aggregate rate requests that the router has approved over the last fraction of a second?

In order to answer this 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], for the current time between T1 and 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 C, and more discussion of these issues is available in [SAF05].)

* If the router's output link has been underutilized and the aggregate 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 8.2 discusses some of the implementation issues in processing Quick-Start requests at routers. [SAF05] 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, [SAF05] also discusses Extreme Quick-Start mechanisms at routers, where the router would keep per-flow state concerning approved Quick-Start requests.

3.5. Quick-Start in IP Tunnels

In this section we consider the effect of IP tunnels on Quick-Start. 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 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.

There are two legitimate ways for handling the Quick-Start Request with IP tunnels:

(1) The tunnel ingress node does not support Quick-Start, or does not approve the Quick-Start request. The node could strip the Quick-Start Request option from the IP header before encapsulation. Alternately, the ingress node can decrement the IP TTL before encapsulation, while leaving the Quick-Start TTL unchanged, thereby changing TTL Diff. This is the assumed behavior of current IP tunnels that are not aware of Quick-Start.

For a tunnel ingress node that does not support Quick-Start, problems with a Quick-Start Request could still occur if a tunnel discards the outer header at egress and does not decrement the inner IP TTL at the ingress. In this case, if both the inner IP TTL and the Quick-Start TTL are decremented after decapsulation at a Quick-Start-aware egress, or if neither is decremented at the egress, then TTL Diff would be the same after egress as it was before ingress, so that it would wrongly appear that all the routers in the tunnel had approved the Quick-Start request. Fortunately, we are not aware of tunnel technologies that operate this way; to the best of our knowledge, all tunnels decrement the IP TTL either at the ingress before encapsulation, or at the egress router after decapsulation, thus changing TTL Diff.

Even the extreme case when the tunnel ingress is at the TCP sender and the tunnel egress is at the TCP receiver, our assumption is that the IP TTL will be decremented either at the tunnel ingress or at the tunnel egress, changing TTL Diff and preventing the end-nodes from wrongly inferring that the Quick-Start Request was approved by all of the routers along the path. If there are tunnels where the IP TTL in not decremented, perhaps for PPP over SSH, then additional attention will have to be paid to the robustness of Quick-Start in these environments.

A Quick-Start-aware egress must also make sure that the Quick-Start Request is not approved if for some reason the inner header includes the Quick-Start Request option, the outer header does not, and the Quick-Start TTL and IP TTL have been decremented in a fashion that makes it appear as if the request has been approved. If the Quick-Start Request doesn't appear in the outer header, then the egress node should remove the Quick-Start Request option from the inner header after decapsulation. Alternately, the egress node could decrement the Rate Request in the Quick-Start Request option to zero.

(2) The tunnel ingress node may choose to support Quick-Start, and locally approve the Quick-Start Request. In this case the IP TTL and Quick-Start option MUST be copied from the inner IP header to the outer header at the tunnel ingress. Upon decapsulation, the IP TTL and the Quick-Start option in the outer IP header MUST be copied back to the inner header. If the ingress router decrements the IP TTL in the inner header before encapsulation, or in the outer header after encapsulation, then if the ingress router wishes to approve the Quick-Start request, it MUST decrement the Quick-Start TTL at the same time, so as not to change TTL Diff. Similarly, if the egress router wishes to approve the Quick-Start request, then when it decrements the IP TTL in the outer header before decapsulation, or in the inner header after decapsulation, it MUST decrement the Quick-Start TTL at the same time.

A tunnel ingress node can support a Quick-Start request without explicitly verifying that the tunnel egress also supports Quick-Start. All that the ingress node has to do is to decrement the IP TTL, but not the Quick-Start TTL, in the inner header after encapsulation. In this case, if the egress node simply discards the outer header at the egress point, TTL Diff will be different after the tunnel egress than it was at the tunnel ingress, and the Quick-Start will not be considered by the end-nodes as having been approved in the network. Thus, the tunnel ingress node on its own can provide protection against egress nodes that might discard the outer header at the egress point.

3.6. A Rate-Reduced Nonce?

One possibility for the Reserved Field, for further investigation, is to use the four bits for a four-bit Rate-Reduced Nonce. The goal of the Rate-Reduced Nonce would be to give the Quick-Start sender some protection against receivers lying about the value of the received Rate Request. The Rate-Reduced Nonce would be initialized by the sender to a random value. When a router approves the Quick-Start request but reduces the Rate Request field, the router resets the Rate-Reduced Nonce to a new random value. When a Quick-Start-capable router denies the Quick-Start request, the router either deletes the Quick-Start Option, or zeroes the Rate-Reduced Nonce when zeroing the Rate Request and the QS TTL. The receiver reports the value of the Rate-Reduced Nonce back to the sender.

The Rate-Reduced Nonce would be of use in cases where the receiver knows the original Rate Request R sent by the sender (e.g., because the sender always uses the same Rate Request), but the Rate Request has been decremented by routers along the path. What prevents the receiver from reporting back to the sender a Rate Request of R, when the received Rate Request was in fact less than R? If the Rate Request was not decremented in the network, then the Rate-Reduced Nonce should have its original value. If the Rate Request *was* decremented in the network, then the probability that the Rate-Reduced Nonce still has its original value is 1/16. Similarly, if the Rate Request was decremented in the network, the chance that the receiver can guess the original value of the Rate-Reduced Nonce is 1/16.

Thus, if the receiver reports back to the sender the original values for the Rate Request and the Rate-Reduced Nonce, and the correct value for the TTL Diff, then it is likely that the Quick-Start Request was in fact approved at its original value by the routers along the path, in particular by all of the Quick-Start-capable routers. The Rate-Reduced Nonce would make it more difficult for the receiver to report that the Rate Request was received at its original value, when in fact the received Rate Request was less than its original value.

We note, however, that the Rate-Reduced Nonce doesn't provide protection against receivers reporting that the Rate Request was decremented by only one step, when it fact it was decremented by many steps in the network. This, if the receiver knows the original Rate Request from the sender, and the received rate request is considerably less than the original request, then the receiver could report a received rate request just one step smaller than the original request, and the Rate-Reduced Nonce wouldn't provide any protection against this.

Section 6.3 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 bytes per second, appropriately formatted, in the Quick-Start Request 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 described above. 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, then host A MUST assume that its Quick-Start request failed. In this case, host A uses TCP's default congestion control procedure. For initial start-up, host A uses the default initial congestion window.

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 packets are defined as packets sent as the result of a successful Quick-Start request, up to the time when the first Quick-Start packet is acknowledged. In order to use Quick-Start, the TCP host MUST use rate-based pacing 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 sent 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. When to Use Quick-Start

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 re-iterate 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 connection initiation when the transport has no idea of the capacity of the network, as discussed above. (A transport that uses TCP Control Block sharing, the Congestion Manager, or the like 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 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 for the appropriate rate at which to send. (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 assess the available capacity on the network path. 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 Rate Request.

Section 4.6 discusses some of the issues of using Quick-Start at connection initiation, and Section 4.7 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

TCP's Quick-Start Response option is defined as follows:

```
       0          1          2          3
+----------+----------+----------+----------+
|   Kind   | Length=4 |   Rate   |   TTL    |
|          |          | Request  |   Diff   |
+----------+----------+----------+----------+
```

```
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 (to be assigned by IANA).

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

The third byte of the Quick-Start Response option contains the allowed Rate Request, formatted as in the Quick-Start 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).

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 Request 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 Quick-Start Response will not be resent if it is lost in the network. Packet loss is 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. If this check is 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 checks of the TTL Diff and the Rate Request are successful, then the TCP 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 the Rate Request in bytes per second, T the measured round-trip time in seconds, and H 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 specified Rate Request. 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 addresses the problem of an overly-large congestion window from an overly-large measurement of the round-trip time.

If the Quick-Start mode ends with all Quick-Start packets being successfully acknowledged, the TCP sender returns to using the default congestion control mechanisms. After all the packets are acknowledged from a Quick-Start request for an initial window, for example, the TCP sender remains in slow-start, if permitted by ssthresh, continuing to increase its congestion window rather aggressively from one round-trip time to the next. 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.

4.5. 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 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, 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. Section A.5 discusses possible alternatives in responding to the loss of a Quick-Start packet.

We note that ECN [RFC3168] can 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.

4.6. 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) determining the rate to request; (2) interactions with Path MTU Discovery; and (3) Quick-Start request packets that are discarded by middleboxes.

4.6.1. Determining the Rate to Request

As discussed in [SAF05], 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. [SAF05] 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.

In the absence of other information, there could be a configured value for the Quick-Start Rate Request. 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. Therefore, it is recommended that the request for the initial sending rate be somewhat conservative, in order to improve the chances for more Quick-Start requests to be approved.

4.6.2. Interactions with Path MTU Discovery

A second 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."

Unfortunately, the sender doesn't necessarily know the Path MTU when it sends packets in the initial window. 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.

One possibility would be for the sender to send one large packet in the initial window with the DF bit set, and to 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, sending the Quick-Start Request with the first window of data while also doing Path MTU Discovery.

In the future, it might be possible for the TCP SYN packet to do a probe about the Path MTU. For example, [W03] has proposed an IP Option that queries routers for their MTU before starting a Path MTU Discovery process.

4.6.3. Quick-Start Request Packets that are Discarded by 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 middleboxes. 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 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].

While RFC 1122 and 2988 recommend that the sender should set the initial RTO to three seconds, many TCP implementations set the initial RTO to one second. For a TCP SYN packet sent with a Quick-Start request, we RECOMMEND an RTO of one second, so that the sender can retransmit the SYN packet reasonably promptly if the original TCP SYN packet is dropped by a middlebox in the network.

In the case of a retransmission, in addition to resending the SYN or SYN/ACK packet without the Quick-Start Request, the TCP sender SHOULD use an RTO of three seconds and a different Initial Sequence Number. Using this scheme the TCP sender MUST keep track of when each of the SYN (or SYN/ACKs) was transmitted. In this way, an acknowledgement for the retransmitted SYN or SYN/ACK packet can be matched with the SYN or SYN/ACK being acknowledged, and the transmission time of the SYN (or SYN/ACK) being acknowledged can be used for an RTT measurement to seed the RTO. If only the retransmitted SYN or SYN/ACK is acknowledged, the TCP sender can reasonably assume that the earlier SYN or SYN/ACK with the Quick-Start option was dropped by the network because of the option and not because of congestion. In this case, the TCP sender can refrain from performing TCP's standard congestion control state changes.

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, to a middlebox dropping the packet because of the IP Option, or because of a 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 IP Options than due to an ECN request in the TCP header [MAF04].

4.7. TCP: A Quick-Start Request in the Middle of 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 connection, for example after an idle period: (1) determining the rate to request; and (2) the response if Quick-Start packets are dropped;

(1) Determining the rate to request:
In the middle of connection, an easy rule of thumb would be for the TCP sender to determine the largest congestion window that the TCP connection achieved since the last packet drop, to translate this congestion window to a sending rate, and use this rate in the Quick-Start request. 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.

In the case of an idle period, the sender SHOULD NOT use Quick-Start if the idle period has been less than an RTO, and the congestion window has not decayed down to less than half of its value at the start of the idle period. Such a use of Quick-Start requires further investigation.

(2) 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 of 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.

We note that a packet in the middle of a connection carrying a Quick-Start Request might or might not carry a data payload. For example, for TCP, the Quick-Start Request could be carried by a data packet, or by a pure acknowledgement packet.

4.8. An Example Quick-Start Scenario with TCP

The following is an example scenario in the case when both hosts request Quick-Start for setting their initial windows:

* 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 and Rate Request 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.

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 and Rate Request 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.

5. 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 and the TTL Diff.

* 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 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).

6. Evaluation of Quick-Start

6.1. 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).

6.2. 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 packet dropped:
For the sender the biggest risk in using Quick-Start lies in the possibility of suffering from congestion-related losses of the Quick-Start 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 has 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 4.6, not all packets would carry the Quick-Start Request 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 6.6, routers can easily protect against this by enforcing a limit on the rate at which Quick-Start requests will be considered.

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. This is, however, similar to what would happen if the connection's path was changed in the middle of the connection, when the connection had already established the allowed initial rate.

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.

6.3. Protection against Misbehaving Nodes

In this section we discuss the protection against receivers or colluding middleboxes lying about the Quick-Start Request. First, we note that it is not necessarily in the receiver's interest to lie about the Quick-Start Request. If the sender sends at too-high of an initial rate, and has a packet dropped, this does not necessarily improve the performance of the connection, relative to the case when the Quick-Start Request was not approved.

6.3.1. 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. If a router that understands the Quick-Start Request denies the request by deleting the request or by zeroing the QS TTL, then the receiver can ‘‘lie" about whether the request was approved only by successfully guessing the value of the TTL Diff to report. The chance of the receiver successfully guessing the correct value for the TTL Diff is 1/256.

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 field, the 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 a TTL field 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.

6.3.2. Receivers Lying about the Approved Rate

A second form of 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 generally doesn't 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. This limits the ability of the receiver to cheat. For example, if the sender sends a Quick-Start Request with an Rate Request of X, and the receiver reports receiving a Quick-Start Request with an 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.

However, if the sender sends a Quick-Start Request with an Rate Request of Z, the receiver receives the Quick-Start Request with an approved Rate Request of X, and reports an Rate Request of Y, for X < Y <= Z, then the receiver succeeds in lying to the sender about the approved rate.

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 forms of protection against receivers lying about the value of the Rate Request. One form of protection would be the Rate-Reduced Nonce discussed earlier, where the receiver would have to report the original value of the nonce if the receiver reported that the original rate request was approved.

A second possible protection would be for a router decreasing 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 third 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.

6.3.3. Collusion between Misbehaving Routers

In addition to protecting against misbehaving receivers, it is necessary also 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 for the request packet at the ingress. This would enable the egress router to modify the QS TTL so that it appeared that all of 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 Request Option from the packet, the ingress router could have sent the Quick-Start Request Option to the egress router out-of-band, with the egress router inserting the Quick-Start Request Option, with a modified QS TTL field, back in the packet.

However, unlike ECN, there is somewhat less 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 of 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 does not necessarily give an advantage to the traffic covered by that collusion. If some router along the path really does not have enough available bandwidth to approve the Quick-Start request, then the Quick-Start packets sent as a result of the falsely-approved request could be dropped in the network, to the resulting 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 necessarily be to the connection's advantage for this to happen. In addition, the router between the ingress and egress nodes that denied the request could be monitoring connection performance, actively penalizing nodes that seem to be using Quick-Start after a Quick-Start request was denied.

If the congested router was ECN-capable, and the colluding ingress and egress routers were 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).

6.3.4. Misbehaving Middleboxes and the IP TTL

A separate possibility is that of traffic normalizers [HKP01] or other middleboxes along that path that re-write IP TTLs, in order to foil other kinds of attacks in the network. If such a traffic normalizer re-wrote 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. Re-writing 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).

6.4. 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. However, we note that routers should be discouraged from granting Quick-Start requests for higher-priority traffic when this is likely to result in significant packet loss for lower-priority traffic.

6.5. Limitations of Quick-Start

The Quick-Start proposal, taken together with HighSpeed TCP [F03], 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. One of the open questions addressed later in this document is whether the limited capabilities of Quick-Start are sufficient to warrant standardization and deployment, or whether more work is needed first to explore the space of potential mechanisms.

6.6. Attacks on Quick-Start

As discussed in [SAF05], Quick-Start is vulnerable to two kinds of Quick-Start 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, attacks to tie up the available Quick-Start bandwidth, is more difficult to defend against. As discussed in [SAF05]. 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 'wasting' 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.

[SAF05] 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-larqe Quick-Start requests.

6.7. 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 4KB and 128KB) in under-utilized 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 [SAF05] 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.

7. Related Work

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 three classes of proposals (no explicit feedback from routers; explicit feedback about the initial rate; and more fine-grained feedback from routers) in the sections below.

7.1. 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 has to have sent some packets already 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 that is possible *with* explicit feedback, it is also possible that approaches that are more aggressive than slow-start are possible without 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. 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.

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:
A second advantage of explicit feedback from the routers is that the available bandwidth along the path does not necessarily map to the allowed sending rate for an individual flow. As an example, if the TCP sender sends four packets back-to-back in the initial window, and the TCP receiver reports that the data packets were received with roughly the same

spacing as they were transmitted, does this mean that the flow can infer an underutilized path? And how fast can the flow send in the next round-trip time? Do the results depend on the level of statistical multiplexing at the congested link, and on the number of flows attempting a faster start-up at the same time?

7.2. 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 key question would be the potential role of optimistic senders in amplifying the damage done by a Distributed Denial of Service (DDoS) attack.

(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.

7.3. 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 Explicit Transport Error Notification (ETEN), which includes a cumulative mechanism to notify endpoints of aggregate congestion statistics along the path [KAPS02].

7.4. 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]. Section A.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.

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.

8. 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.

8.1. Implementation Issues for Sending Quick-Start Requests

Section 4.6 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 the 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 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. Datagram-based real-time streaming applications, on the other hand, may have a specific preference on the transmission rate and they could indicate the required rate explicitly to the transport protocol to be used in the Quick-Start Request.

We note that when Quick-Start is used, the TCP sender is required to implement an additional timer for the paced transmission of Quick-Start packets.

8.2. 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 network 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 send end up being dropped.

8.3. Possible Deployment Scenarios

Because of possible problems discussed above concerning using Quick-Start over some network paths, the most realistic initial deployment of Quick-Start would likely to 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.

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 often have large network capacity and the networks are underutilized for most of the time. 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.

* 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 are typically up to one 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.

* 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.

8.4. Would QuickStart packets take the slow path in routers?

How much delay would the slow path add to the processing time for this packet? Similarly, if QuickStart packets took the slow path, how much stress would it add to routers for there to be many more packets on the slow path, because of the number of packets using QuickStart? These are both questions to be explored while experimenting with Quick-Start in the Internet.

8.5. 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 its 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.

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.

9. Security Considerations

Sections 6.3 and 6.6 discuss the security considerations related to Quick-Start. Section 6.3 discusses the potential abuse of Quick-Start of receivers lying about whether the request was approved or about the approved rate; of routers in collusion to misuse Quick-Start; and of potential problems with traffic normalizers that rewrite IP TTLs in packet headers. All of these problems could result in the sender using an 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 would 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 6.6 discusses a potential attack on the routers' processing and state load from an attack of Quick-Start Requests. Section 6.6 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.

Section 4.6.3 discusses the potential problem of packets with Quick-Start Requests dropped by middleboxes along the path.

10. Conclusions

We are presenting the Quick-Start mechanism as a proposal for a simple, understandable, and incrementally-deployable mechanism that would be sufficient to allow connections to start up with large initial rates, or large initial congestion windows, in overprovisioned, high-bandwidth environments. We expect there will be an increasing number of overprovisioned, 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.

11. 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. We also thank Mohammed Ashraf, John Border, Tom Dunigan, John Heidemann, Paul Hyder, Dina Katabi, and Vern Paxson for feedback. This draft builds upon the concepts described in [RFC3390], [AHO98], [RFC2415], and [RFC3168].

This is a modification of a draft originally by Amit Jain for Initial Window Discovery.

A. Design Decisions

A.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, RFC2463] or RSVP [RFC2205] protocols could be used for delivering the Quick-Start Request.

A.1.1. ICMP

Being a control protocol used between Internet nodes, one could argue that ICMP is the ideal method for requesting a permission for faster startup 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; for now, we don't address the mechanisms that would be needed to accomplish this task. 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. If the ICMP Quick-Start Request reached the receiver, the receiver would use transport-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 was sent concurrently 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 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 tunnel mode [RFC2401]. 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.

A.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 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.

A.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 both four-bit and eight-bit Rate Request fields.

Linear functions:
The Quick-Start Request contains an 8-bit field for the Rate Request. One possible proposal would be for this field to be formatted in bits per second, scaled so that one unit equals 80 Kbps. Thus, for the value N in the Rate Request field, the requested rate is 80,000*N bps. This gives a request range between 80 Kbps and 20.48 Mbps. For 1500-byte packets, this corresponds to a request range between 6 and 1706 packets per second.

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, the request range would be from 80 Kbps to 40*2^256 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 is possible 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 is not acceptable, then five bits could be used for the Rate Request field.

If the granularity of factors of two is 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.

We note that the Rate Request also has to be 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.

A.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 will 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 draft 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. 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 an Rate Request field in bytes per second rather than in packets per second because it seems somewhat more robust, particularly to routers.

A.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.

For either of these alternatives, there would not be room to report the current sending rate in the Quick-Start Option using the current minimal format for the Quick-Start Request. Thus, either the Quick-Start Option would have to take more than four bytes to include a report of the current sending rate, or the current sending rate would not be reported to the routers.

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

Section 4.5 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. 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 any such approaches could give the TCP receiver an incentive to lie about the Quick-Start request. That is, if the sender reverts to slow-start when a Quick-Start packet is dropped, then it is generally not to the receiver's advantage to report a larger rate request than was actually approved if the result is going to be a Quick-Start packet dropped in the network. However, if the receiver benefits from a larger Quick-Start window even when the larger window results in Quick-Start packets dropped in the network, then the receiver has a greater incentive to lie about the received rate request, in an effort to get the sender to use a larger initial sending rate.

A.6. Why Not Include More Functionality?

As Section 6.5 discussed, this proposal for Quick-Start is a rather coarse-grained mechanism that would allow connections to use higher sending rates along underutilized paths, but that does not attempt to provide a next-generation transport protocol, and does not attempt the goal of providing very high throughput with very low delay. As Section 7.4 discusses, there are a number of proposals such as XCP, MaxNet, and AntiECN for more fine-grained per-packet feedback from routers that the current congestion control mechanisms, that do 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 of the routers along the path are significantly under-utilized. 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, 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, and (we think) 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 startups 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 multi-path 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 travelled on path A, while half of the Quick-Start packets sent in the succeeding round-trip time are routed on path B.

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. For example, for a mechanism for requesting a initial sending rate in an underutilized environment, the fairness issues of a general congestion control mechanism go away, and 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 2 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 2: 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 of meeting some of the 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.

A.7. The Earlier QuickStart Nonce

An earlier version of this document included a Request-Approved QuickStart 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 QuickStart Nonce would have been returned by the TCP receiver to the TCP 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 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 field when they deny a request.

With the old Request-Approved QuickStart 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 single field for the QS TTL, instead of two fields, one for the QS TTL and another for a QS-Approved Nonce.

The Quick-Start Nonce has been resurrected in the current proposal for a Rate-Reduced Nonce given in Section 3.6. This proposal would provide specific protection against receivers lying about whether the rate request was decremented by routers along the path. In this proposal, the Rate-Reduced Nonce would be reset to a new random value by routers that approve the request but decrement the value of the Rate Request. Thus, if the original value for the Rate-Reduced Nonce is reported back by the receiver to the sender, then it is likely that the Rate Request was not decremented or denied by Quick-Start-capable routers along the path.

B. 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 on-line 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 TFRC, an equation-based form of congestion control. We refer the reader to [KHF05] for a more detailed description of DCCP, and of the 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; and (2) when does the sender determine that there has been no feedback from the receiver, and reduce the sending rate?

(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 prudent to use a smaller time interval, e.g., to reduce the sending rate if no feedback is received 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 sufficient; 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; (2) what is the response to a loss; and (3) when does the sender determine that there has been no feedback from the receiver, and the sending rate must be reduced?

(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 that 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.

C. 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.

[SAF05] 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 [SAF05]). 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 [SAF05] 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 [SAF05]). 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 simpliest method, outlined in Section 3.4, 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 [SAF05] 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 is less than the threshold then Rate Requests will 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 gross the approved rate should be rounded down when it does not fall exactly on one of the rates allowed by the encoding scheme.

Normative References

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

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

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

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

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

[RFC3390] M. Allman, S. Floyd, 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, Experimental, March 2004.

Informative References

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

[RFC1812] F. Baker (ed.). Requirements for IP Version 4 Routers. RFC 1812, June 1995.

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

[RFC2205] R. Braden, et al. Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification. RFC 2205, September 1997.

[RFC2309] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April 1998.

[RFC2401] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, November 1998.

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

[RFC2416] T. Shepard and C. Partridge. When TCP Starts Up With Four Packets Into

Only Three Buffers. RFC 2416. September 1998.

[RFC2463] A. Conta and S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 2463, December 1998.

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

[RFC2960] R. Stewart, et. al. Stream Control Transmission Protocol. RFC 2960, October 2000.

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

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

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

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

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

[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.

[FF99] Floyd, S., and Fall, K., Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, August 1999.

[F03] Floyd, S., HighSpeed TCP for Large Congestion Windows, RFC 3649, December 2003.

[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.

[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.

[KHR02] Dina Katabi, Mark Handley, and Charles Rohrs, Internet Congestion Control for

Future High Bandwidth-Delay Product Environments. ACM Sigcomm 2002, August 2002. URL "http://ana.lcs.mit.edu/dina/XCP/".

[KHF05] E. Kohler, M. Handley, and S. Floyd, Datagram Congestion Control Protocol (DCCP), internet draft draft-ietf-dccp-spec-11.txt, work in progress, March 2005.

[K03] S. Kunniyur, "AntiECN Marking: A Marking Scheme for High Bandwidth Delay Connections", Proceedings, IEEE ICC ’03, May 2003. URL "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. URL "http://www.icir.org/mallman/papers/".

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

[MAF05] Alberto Medina, Mark Allman, and Sally Floyd. Measuring the Evolution of Transport Protocols in the Internet. To appear in Computer Communications Review, April 2004.

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

[PK98] Venkata N. Padmanabhan and Randy H. Katz, TCP Fast Start: A Technique For Speeding Up Web Transfers, IEEE GLOBECOM ’98, November 1998.

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

[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. URL "http://www.icir.org/mallman/papers/".

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

[SAF05] Pasi Sarolahti, Mark Allman, and Sally Floyd. Evaluating Quick-Start for TCP. Under submission, May 2005. URL "http://www.icir.org/floyd/quickstart.html".

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

[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. URL "http://informatik.uibk.ac.at/users/c70370/research/publications/".

[W03] Michael Welzl, PMTU-Options: Path MTU Discovery Using Options, expired internet-draft draft-welzl-pmtud-options-01.txt, work-in-progress. February 2003.

[ZPS00] Y. Zhang, V. Paxson, and S. Shenker, The Stationarity of Internet Path Properties: Routing, Loss, and Throughput, ACIRI Technical Report, May 2000.

IANA Considerations

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

IP Option

Quick-Start requires an IP Option Number to be allocated. The IP Option would have a copied flag of 0, a class field of 00, and the assigned five-bit option number. The name of the option would be "QSR - Quick-Start Request", with this document as the reference document.

TCP Option

Quick-Start also requires that a TCP Option Number be allocated. The Length would be 4, and the Meaning would be "Quick-Start Request", with this document as the reference document.

AUTHORS' ADDRESSES

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

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) 243-7361
Email: mallman@icir.org
URL: http://www.icir.org/mallman/

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

Full Copyright Statement

Intellectual Property