HighSpeed TCP for Large Congestion Windows

Status of this Memo

Copyright Notice

Abstract

   The proposals in this document are experimental. While they may be
   deployed in the current Internet, they do not represent a consensus
   that this is the best method for high-speed congestion control. In
   particular, we note that alternative experimental proposals are
   likely to be forthcoming, and it is not well understood how the
   proposals in this document will interact with such alternative
   proposals.

   This document proposes HighSpeed TCP, a modification to TCP's
   congestion control mechanism for use with TCP connections with large
   congestion windows. The congestion control mechanisms of the current
   Standard TCP constrains the congestion windows that can be achieved
   by TCP in realistic environments. For example, for a Standard TCP
   connection with 1500-byte packets and a 100 ms round-trip time,
   achieving a steady-state throughput of 10 Gbps would require an
   average congestion window of 83,333 segments, and a packet drop rate
   of at most one congestion event every 5,000,000,000 packets (or
   equivalently, at most one congestion event every 1 2/3 hours). This
   is widely acknowledged as an unrealistic constraint. To address this
   limitation of TCP, this document proposes HighSpeed TCP, and solicits
   experimentation and feedback from the wider community.

Table of Contents

1. Introduction

   This document proposes HighSpeed TCP, a modification to TCP's
   congestion control mechanism for use with TCP connections with large
   congestion windows. In a steady-state environment, with a packet
   loss rate p, the current Standard TCP's average congestion window is
   roughly $1.2/\sqrt{p}$ segments. This places a serious constraint on
   the congestion windows that can be achieved by TCP in realistic
   environments. For example, for a Standard TCP connection with 1500-
   byte packets and a 100 ms round-trip time, achieving a steady-state
   throughput of 10 Gbps would require an average congestion window of

83,333 segments, and a packet drop rate of at most one congestion
event every 5,000,000,000 packets (or equivalently, at most one
congestion event every 1 2/3 hours). The average packet drop rate of
at most $2*10^{(-10)}$ needed for full link utilization in this
environment corresponds to a bit error rate of at most $2*10^{(-14)}$,
and this is an unrealistic requirement for current networks.

To address this fundamental limitation of TCP and of the TCP response
function (the function mapping the steady-state packet drop rate to
TCP's average sending rate in packets per round-trip time), this
document describes a modified TCP response function for regimes with
higher congestion windows. This document also solicits
experimentation and feedback on HighSpeed TCP from the wider
community.

Because HighSpeed TCP's modified response function would only take
effect with higher congestion windows, HighSpeed TCP does not modify
TCP behavior in environments with heavy congestion, and therefore
does not introduce any new dangers of congestion collapse. However,
if relative fairness between HighSpeed TCP connections is to be
preserved, then in our view any modification to the TCP response
function should be addressed in the IETF, rather than made as ad hoc
decisions by individual implementors or TCP senders. Modifications
to the TCP response function would also have implications for
transport protocols that use TFRC and other forms of equation-based
congestion control, as these congestion control mechanisms directly
use the TCP response function [RFC3448].

This proposal for HighSpeed TCP focuses specifically on a proposed
change to the TCP response function, and its implications for TCP.
This document does not address what we view as a separate fundamental
issue, of the mechanisms required to enable best-effort connections
to *start* with large initial windows. In our view, while HighSpeed
TCP proposes a somewhat fundamental change to the TCP response
function, at the same time it is a relatively simple change to
implement in a single TCP sender, and presents no dangers in terms of
congestion collapse. In contrast, in our view, the problem of
enabling connections to *start* with large initial windows is
inherently more risky and structurally more difficult, requiring some
form of explicit feedback from all of the routers along the path.
This is another reason why we would propose addressing the problem of
starting with large initial windows separately, and on a separate
timetable, from the problem of modifying the TCP response function.

2. The Problem Description

   This section describes the number of round-trip times between
   congestion events required for a Standard TCP flow to achieve an
   average throughput of B bps, given packets of D bytes and a round-
   trip time of R seconds. A congestion event refers to a window of
   data with one or more dropped or ECN-marked packets (where ECN stands
   for Explicit Congestion Notification).

   From Appendix A, achieving an average TCP throughput of B bps
   requires a loss event at most every $BR/(12D)$ round-trip times. This
   is illustrated in Table 1, for R = 0.1 seconds and D = 1500 bytes.
   The table also gives the average congestion window W of $BR/(8D)$, and
   the steady-state packet drop rate P of $1.5/W^2$.

TCP Throughput (Mbps)	RTTs Between Losses	W	P
1	5.5	8.3	0.02
10	55.5	83.3	0.0002
100	555.5	833.3	0.000002
1000	5555.5	8333.3	0.00000002
10000	55555.5	83333.3	0.0000000002

   Table 1: RTTs Between Congestion Events for Standard TCP, for
   1500-Byte Packets and a Round-Trip Time of 0.1 Seconds.

   This document proposes HighSpeed TCP, a minimal modification to TCP's
   increase and decrease parameters, for TCP connections with larger
   congestion windows, to allow TCP to achieve high throughput with more
   realistic requirements for the steady-state packet drop rate.
   Equivalently, HighSpeed TCP has more realistic requirements for the
   number of round-trip times between loss events.

3. Design Guidelines

   Our proposal for HighSpeed TCP is motivated by the following
   requirements:

   * Achieve high per-connection throughput without requiring
      unrealistically low packet loss rates.

   * Reach high throughput reasonably quickly when in slow-start.

   * Reach high throughput without overly long delays when recovering
      from multiple retransmit timeouts, or when ramping-up from a
      period with small congestion windows.

   * No additional feedback or support required from routers:

   For example, the goal is for acceptable performance in both ECN-
   capable and non-ECN-capable environments, and with Drop-Tail as well
   as with Active Queue Management such as RED in the routers.

   * No additional feedback required from TCP receivers.

   * TCP-compatible performance in environments with moderate or high
      congestion (e.g., packet drop rates of 1% or higher):

   Equivalently, the requirement is that there be no additional load on
   the network (in terms of increased packet drop rates) in environments
   with moderate or high congestion.

   * Performance at least as good as Standard TCP in environments with
      moderate or high congestion.

   * Acceptable transient performance, in terms of increases in the
      congestion window in one round-trip time, responses to severe
      congestion, and convergence times to fairness.

   Currently, users wishing to achieve throughputs of 1 Gbps or more
   typically open up multiple TCP connections in parallel, or use MulTCP
   [CO98,GRK99], which behaves roughly like the aggregate of N virtual
   TCP connections. While this approach suffices for the occasional
   user on well-provisioned links, it leaves the parameter N to be
   determined by the user, and results in more aggressive performance
   and higher steady-state packet drop rates if used in environments
   with periods of moderate or high congestion. We believe that a new
   approach is needed that offers more flexibility, more effectively
   scales to a wide range of available bandwidths, and competes more
   fairly with Standard TCP in congested environments.

4. Non-Goals

   The following are explicitly *not* goals of our work:

   * Non-goal: TCP-compatible performance in environments with very low
      packet drop rates.

   We note that our proposal does not require, or deliver, TCP-
   compatible performance in environments with very low packet drop
   rates, e.g., with packet loss rates of 10^{-5} or 10^{-6}. As we discuss
   later in this document, we assume that Standard TCP is unable to make
   effective use of the available bandwidth in environments with loss

rates of 10^-6 in any case, so that it is acceptable and appropriate
for HighSpeed TCP to perform more aggressively than Standard TCP in
such an environment.

* Non-goal: Ramping-up more quickly than allowed by slow-start.

It is our belief that ramping-up more quickly than allowed by slow-
start would necessitate more explicit feedback from routers along the
path. The proposal for HighSpeed TCP is focused on changes to TCP
that could be effectively deployed in the current Internet
environment.

* Non-goal: Avoiding oscillations in environments with only one-way,
   long-lived flows all with the same round-trip times.

While we agree that attention to oscillatory behavior is useful,
avoiding oscillations in aggregate throughput has not been our
primary consideration, particularly for simplified environments
limited to one-way, long-lived flows all with the same, large round-
trip times. Our assessment is that some oscillatory behavior in
these extreme environments is an acceptable price to pay for the
other benefits of HighSpeed TCP.

5. Modifying the TCP Response Function

The TCP response function, $w = 1.2/\sqrt{p}$, gives TCP's average
congestion window w in MSS-sized segments, as a function of the
steady-state packet drop rate p [FF98]. This TCP response function
is a direct consequence of TCP's Additive Increase Multiplicative
Decrease (AIMD) mechanisms of increasing the congestion window by
roughly one segment per round-trip time in the absence of congestion,
and halving the congestion window in response to a round-trip time
with a congestion event. This response function for Standard TCP is
reflected in the table below. In this proposal we restrict our
attention to TCP performance in environments with packet loss rates
of at most 10^-2, and so we can ignore the more complex response
functions that are required to model TCP performance in more
congested environments with retransmit timeouts. From Appendix A, an
average congestion window of W corresponds to an average of 2/3 W
round-trip times between loss events for Standard TCP (with the
congestion window varying from 2/3 W to 4/3 W).

Packet Drop Rate P	Congestion Window W	RTTs Between Losses
10^{-2}	12	8
10^{-3}	38	25
10^{-4}	120	80
10^{-5}	379	252
10^{-6}	1200	800
10^{-7}	3795	2530
10^{-8}	12000	8000
10^{-9}	37948	25298
10^{-10}	120000	80000

Table 2: TCP Response Function for Standard TCP. The average
congestion window W in MSS-sized segments is given as a function of
the packet drop rate P.

To specify a modified response function for HighSpeed TCP, we use
three parameters, Low_Window, High_Window, and High_P. To ensure TCP
compatibility, the HighSpeed response function uses the same response
function as Standard TCP when the current congestion window is at
most Low_Window, and uses the HighSpeed response function when the
current congestion window is greater than Low_Window. In this
document we set Low_Window to 38 MSS-sized segments, corresponding to
a packet drop rate of 10^{-3} for TCP.

To specify the upper end of the HighSpeed response function, we
specify the packet drop rate needed in the HighSpeed response
function to achieve an average congestion window of 83000 segments.
This is roughly the window needed to sustain 10 Gbps throughput, for
a TCP connection with the default packet size and round-trip time
used earlier in this document. For High_Window set to 83000, we
specify High_P of 10^{-7}; that is, with HighSpeed TCP a packet drop
rate of 10^{-7} allows the HighSpeed TCP connection to achieve an
average congestion window of 83000 segments. We believe that this
loss rate sets an achievable target for high-speed environments,
while still allowing acceptable fairness for the HighSpeed response
function when competing with Standard TCP in environments with packet
drop rates of 10^{-4} or 10^5.

For simplicity, for the HighSpeed response function we maintain the
property that the response function gives a straight line on a log-
log scale (as does the response function for Standard TCP, for low to
moderate congestion). This results in the following response
function, for values of the average congestion window W greater than
Low_Window:

  $W = (p/Low\_P)^S Low\_Window,$

for Low_P the packet drop rate corresponding to Low_Window, and for S
as following constant [FRS02]:

  S = (log High_Window - log Low_Window)/(log High_P - log Low_P).

(In this paper, "log x" refers to the log base 10.) For example, for
Low_Window set to 38, we have Low_P of 10^-3 (for compatibility with
Standard TCP). Thus, for High_Window set to 83000 and High_P set to
10^-7, we get the following response function:

  W = 0.12/p^0.835.   (1)

This HighSpeed response function is illustrated in Table 3 below.
For HighSpeed TCP, the number of round-trip times between losses,
1/(pW), equals 12.7 W^0.2, for W > 38 segments.

Packet Drop Rate P	Congestion Window W	RTTs Between Losses
10^-2	12	8
10^-3	38	25
10^-4	263	38
10^-5	1795	57
10^-6	12279	83
10^-7	83981	123
10^-8	574356	180
10^-9	3928088	264
10^-10	26864653	388

Table 3: TCP Response Function for HighSpeed TCP. The average
congestion window W in MSS-sized segments is given as a function of
the packet drop rate P.

We believe that the problem of backward compatibility with Standard
TCP requires a response function that is quite close to that of
Standard TCP for loss rates of 10^-1, 10^-2, or 10^-3. We believe,
however, that such stringent TCP-compatibility is not required for
smaller loss rates, and that an appropriate response function is one
that gives a plausible packet drop rate for a connection throughput
of 10 Gbps. This also gives a slowly increasing number of round-trip
times between loss events as a function of a decreasing packet drop
rate.

Another way to look at the HighSpeed response function is to consider
that HighSpeed TCP is roughly emulating the congestion control
response of N parallel TCP connections, where N is initially one, and
where N increases as a function of the HighSpeed TCP's congestion
window. Thus for the HighSpeed response function in Equation (1)
above, the response function can be viewed as equivalent to that of

N(W) parallel TCP connections, where N(W) varies as a function of the
congestion window W. Recall that for a single standard TCP
connection, the average congestion window equals 1.2/sqrt(p). For N
parallel TCP connections, the aggregate congestion window for the N
connections equals N*1.2/sqrt(p). From the HighSpeed response
function in Equation (1) and the relationship above, we can derive
the following:

    N(W) = 0.23*W^(0.4)

for N(W) the number of parallel TCP connections emulated by the
HighSpeed TCP response function, and for N(W) >= 1. This is shown in
Table 4 below.

Congestion Window W	Number N(W) of Parallel TCPs
1	1
10	1
100	1.4
1,000	3.6
10,000	9.2
100,000	23.0

Table 4: Number N(W) of parallel TCP connections roughly emulated by
the HighSpeed TCP response function.

In this document, we do not attempt to seriously evaluate the
HighSpeed response function for congestion windows greater than
100,000 packets. We believe that we will learn more about the
requirements for sustaining the throughput of best-effort connections
in that range as we gain more experience with HighSpeed TCP with
congestion windows of thousands and tens of thousands of packets.
There also might be limitations to the per-connection throughput that
can be realistically achieved for best-effort traffic, in terms of
congestion window of hundreds of thousands of packets or more, in the
absence of additional support or feedback from the routers along the
path.

6. Fairness Implications of the HighSpeed Response Function

    The Standard and Highspeed Response Functions can be used directly to
    infer the relative fairness between flows using the two response
    functions. For example, given a packet drop rate P, assume that
    Standard TCP has an average congestion window of W_Standard, and
    HighSpeed TCP has a higher average congestion window of W_HighSpeed.

In this case, a single HighSpeed TCP connection is receiving
W_HighSpeed/W_Standard times the throughput of a single Standard TCP
connection competing in the same environment.

This relative fairness is illustrated below in Table 5, for the
parameters used for the Highspeed response function in the section
above. The second column gives the relative fairness, for the
steady-state packet drop rate specified in the first column. To help
calibrate, the third column gives the aggregate average congestion
window for the two TCP connections, and the fourth column gives the
bandwidth that would be needed by the two connections to achieve that
aggregate window and packet drop rate, given 100 ms round-trip times
and 1500-byte packets.

```
    Packet Drop Rate P   Fairness  Aggregate Window  Bandwidth
    ------------------   --------  ----------------  ---------
            10^-2          1.0                 24      2.8 Mbps
            10^-3          1.0                 76      9.1 Mbps
            10^-4          2.2                383     45.9 Mbps
            10^-5          4.7               2174    260.8 Mbps
            10^-6         10.2              13479      1.6 Gbps
            10^-7         22.1              87776     10.5 Gbps
```

Table 5: Relative Fairness between the HighSpeed and Standard
Response Functions.

Thus, for packet drop rates of 10^-4, a flow with the HighSpeed
response function can expect to receive 2.2 times the throughput of a
flow using the Standard response function, given the same round-trip
times and packet sizes. With packet drop rates of 10^-6 (or 10^-7),
the unfairness is more severe, and we have entered the regime where a
Standard TCP connection requires at most one congestion event every
800 (or 2530) round-trip times in order to make use of the available
bandwidth. Our judgement would be that there are not a lot of TCP
connections effectively operating in this regime today, with
congestion windows of thousands of packets, and that therefore the
benefits of the HighSpeed response function would outweigh the
unfairness that would be experienced by Standard TCP in this regime.
However, one purpose of this document is to solicit feedback on this
issue. The parameter Low_Window determines directly the point of
divergence between the Standard and HighSpeed Response Functions.

The third column of Table 5, the Aggregate Window, gives the
aggregate congestion window of the two competing TCP connections,
with HighSpeed and Standard TCP, given the packet drop rate specified
in the first column. From Table 5, a HighSpeed TCP connection would
receive ten times the bandwidth of a Standard TCP in an environment
with a packet drop rate of 10^-6. This would occur when the two

flows sharing a single pipe achieved an aggregate window of 13479
packets. Given a round-trip time of 100 ms and a packet size of 1500
bytes, this would occur with an available bandwidth for the two
competing flows of 1.6 Gbps.

Next we consider the time that it takes a standard or HighSpeed TCP
flow to converge to fairness against a pre-existing HighSpeed TCP
flow. The worst case for convergence to fairness occurs when a new
flow is starting up, competing against a high-bandwidth existing
flow, and the new flow suffers a packet drop and exits slow-start
while its window is still small. In the worst case, consider that
the new flow has entered the congestion avoidance phase while its
window is only one packet. A standard TCP flow in congestion
avoidance increases its window by at most one packet per round-trip
time, and after N round-trip times has only achieved a window of N
packets (when starting with a window of 1 in the first round-trip
time). In contrast, a HighSpeed TCP flows increases much faster than
a standard TCP flow while in the congestion avoidance phase, and we
can expect its convergence to fairness to be much better. This is
shown in Table 6 below. The script used to generate this table is
given in Appendix C.

```
  RTT   HS_Window Standard_TCP_Window
  ---   --------- -------------------
  100       131         100
  200       475         200
  300      1131         300
  400      2160         400
  500      3601         500
  600      5477         600
  700      7799         700
  800     10567         800
  900     13774         900
 1000     17409        1000
 1100     21455        1100
 1200     25893        1200
 1300     30701        1300
 1400     35856        1400
 1500     41336        1500
 1600     47115        1600
 1700     53170        1700
 1800     59477        1800
 1900     66013        1900
 2000     72754        2000
```

Table 6: For a HighSpeed and a Standard TCP connection, the
congestion window during congestion avoidance phase (starting with a
congestion window of 1 packet during RTT 1).

The classic paper on relative fairness is from Chiu and Jain [CJ89].
This paper shows that AIMD (Additive Increase Multiplicative
Decrease) converges to fairness in an environment with synchronized
congestion events. From [CJ89], it is easy to see that MIMD and AIAD
do not converge to fairness in this environment. However, the
results of [CJ89] do not apply to an asynchronous environment such as
that of the current Internet, where the frequency of congestion
feedback can be different for different flows. For example, it has
been shown that MIMD converges to fair states in a model with
proportional instead of synchronous feedback in terms of packet drops
[GV02]. Thus, we are not concerned about abandoning a strict model
of AIMD for HighSpeed TCP. However, we note that in an environment
with Drop-Tail queue management, there is likely to be some
synchronization of packet drops. In this environment, the model of
completely synchronous feedback does not hold, but the model of
completely asynchronous feedback is not accurate either. Fairness in
Drop-Tail environments is discussed in more detail in Sections 9 and
12.

7. Translating the HighSpeed Response Function into Congestion Control
    Parameters

For equation-based congestion control such as TFRC, the HighSpeed
Response Function above could be used directly by the TFRC congestion
control mechanism. However, for TCP the HighSpeed response function
has to be translated into additive increase and multiplicative
decrease parameters. The HighSpeed response function cannot be
achieved by TCP with an additive increase of one segment per round-
trip time and a multiplicative decrease of halving the current
congestion window; HighSpeed TCP will have to modify either the
increase or the decrease parameter, or both. We have concluded that
HighSpeed TCP is most likely to achieve an acceptable compromise
between moderate increases and timely decreases by modifying both the
increase and the decrease parameter.

That is, for HighSpeed TCP let the congestion window increase by $a(w)$
segments per round-trip time in the absence of congestion, and let
the congestion window decrease to $w(1-b(w))$ segments in response to a
round-trip time with one or more loss events. Thus, in response to a
single acknowledgement HighSpeed TCP increases its congestion window
in segments as follows:

 $w \leftarrow w + a(w)/w$.

In response to a congestion event, HighSpeed TCP decreases as
follows:

 $w \leftarrow (1-b(w))w$.

For Standard TCP, a(w) = 1 and b(w) = 1/2, regardless of the value of
w. HighSpeed TCP uses the same values of a(w) and b(w) for w <=
Low_Window. This section specifies a(w) and b(w) for HighSpeed TCP
for larger values of w.

For w = High_Window, we have specified a loss rate of High_P. From
[FRS02], or from elementary calculations, this requires the following
relationship between a(w) and b(w) for w = High_Window:

  a(w) = High_Window^2 * High_P * 2 * b(w)/(2-b(w)).   (2)

We use the parameter High_Decrease to specify the decrease parameter
b(w) for w = High_Window, and use Equation (2) to derive the increase
parameter a(w) for w = High_Window. Along with High_P = 10^-7 and
High_Window = 83000, for example, we specify High_Decrease = 0.1,
specifying that b(83000) = 0.1, giving a decrease of 10% after a
congestion event. Equation (2) then gives a(83000) = 72, for an
increase of 72 segments, or just under 0.1%, within a round-trip
time, for w = 83000.

This moderate decrease strikes us as acceptable, particularly when
coupled with the role of TCP's ACK-clocking in limiting the sending
rate in response to more severe congestion [BBFS01]. A more severe
decrease would require a more aggressive increase in the congestion
window for a round-trip time without congestion. In particular, a
decrease factor High_Decrease of 0.5, as in Standard TCP, would
require an increase of 459 segments per round-trip time when w =
83000.

Given decrease parameters of b(w) = 1/2 for w = Low_Window, and b(w)
= High_Decrease for w = High_Window, we are left to specify the value
of b(w) for other values of w > Low_Window. From [FRS02], we let
b(w) vary linearly as the log of w, as follows:

  b(w) = (High_Decrease - 0.5) (log(w)-log(W)) / (log(W_1)-log(W)) +
  0.5,

for W = Low_window and W_1 = High_window. The increase parameter
a(w) can then be computed as follows:

  a(w) = w^2 * p(w) * 2 * b(w)/(2-b(w)),

for p(w) the packet drop rate for congestion window w. From
inverting Equation (1), we get p(w) as follows:

  p(w) = 0.078/w^1.2.

We assume that experimental implementations of HighSpeed TCP for
further investigation will use a pre-computed look-up table for
finding a(w) and b(w). For example, the implementation from Tom
Dunigan adjusts the a(w) and b(w) parameters every 0.1 seconds. In
the appendix we give such a table for our default values of
Low_Window = 38, High_Window = 83,000, High_P = 10^-7, and
High_Decrease = 0.1. These are also the default values in the NS
simulator; example simulations in NS can be run with the command
"./test-all-tcpHighspeed" in the directory tcl/test.

8. An alternate, linear response functions

   In this section we explore an alternate, linear response function for
   HighSpeed TCP that has been proposed by a number of other people, in
   particular by Glenn Vinnicombe and Tom Kelly. Similarly, it has been
   suggested by others that a less "ad-hoc" guideline for a response
   function for HighSpeed TCP would be to specify a constant value for
   the number of round-trip times between congestion events.

   Assume that we keep the value of Low_Window as 38 MSS-sized segments,
   indicating when the HighSpeed response function diverges from the
   current TCP response function, but that we modify the High_Window and
   High_P parameters that specify the upper range of the HighSpeed
   response function. In particular, consider the response function
   given by High_Window = 380,000 and High_P = 10^-7, with Low_Window =
   38 and Low_P = 10^-3 as before.

   Using the equations in Section 5, this would give the following
   Linear response function, for w > Low_Window:

   $$W = 0.038/p.$$

   This Linear HighSpeed response function is illustrated in Table 7
   below. For HighSpeed TCP, the number of round-trip times between
   losses, 1/(pW), equals 1/0.38, or equivalently, 26, for W > 38
   segments.

Packet Drop Rate P	Congestion Window W	RTTs Between Losses
10^-2	12	8
10^-3	38	26
10^-4	380	26
10^-5	3800	26
10^-6	38000	26
10^-7	380000	26
10^-8	3800000	26
10^-9	38000000	26
10^-10	380000000	26

Table 7: An Alternate, Linear TCP Response Function for HighSpeed
TCP. The average congestion window W in MSS-sized segments is given
as a function of the packet drop rate P.

Given a constant decrease b(w) of 1/2, this would give an increase
a(w) of w/Low_Window, or equivalently, a constant increase of
1/Low_Window packets per acknowledgement, for w > Low_Window.
Another possibility is Scalable TCP [K03], which uses a fixed
decrease b(w) of 1/8 and a fixed increase per acknowledgement of
0.01. This gives an increase a(w) per window of 0.005 w, for a TCP
with delayed acknowledgements, for pure MIMD.

The relative fairness between the alternate Linear response function
and the standard TCP response function is illustrated below in Table
8.

Packet Drop Rate P	Fairness	Aggregate Window	Bandwidth
10^-2	1.0	24	2.8 Mbps
10^-3	1.0	76	9.1 Mbps
10^-4	3.2	500	60.0 Mbps
10^-5	15.1	4179	501.4 Mbps
10^-6	31.6	39200	4.7 Gbps
10^-7	100.1	383795	46.0 Gbps

Table 8: Relative Fairness between the Linear HighSpeed and Standard
Response Functions.

One attraction of the linear response function is that it is scale-
invariant, with a fixed increase in the congestion window per
acknowledgement, and a fixed number of round-trip times between loss
events. My own assumption would be that having a fixed length for
the congestion epoch in round-trip times, regardless of the packet
drop rate, would be a poor fit for an imprecise and imperfect world
with routers with a range of queue management mechanisms, such as the
Drop-Tail queue management that is common today. For example, a

response function with a fixed length for the congestion epoch in
round-trip times might give less clearly-differentiated feedback in
an environment with steady-state background losses at fixed intervals
for all flows (as might occur with a wireless link with occasional
short error bursts, giving losses for all flows every N seconds
regardless of their sending rate).

While it is not a goal to have perfect fairness in an environment
with synchronized losses, it would be good to have moderately
acceptable performance in this regime. This goal might argue against
a response function with a constant number of round-trip times
between congestion events. However, this is a question that could
clearly use additional research and investigation. In addition,
flows with different round-trip times would have different time
durations for congestion epochs even in the model with a linear
response function.

The third column of Table 8, the Aggregate Window, gives the
aggregate congestion window of two competing TCP connections, one
with Linear HighSpeed TCP and one with Standard TCP, given the packet
drop rate specified in the first column. From Table 8, a Linear
HighSpeed TCP connection would receive fifteen times the bandwidth of
a Standard TCP in an environment with a packet drop rate of 10^{-5}.
This would occur when the two flows sharing a single pipe achieved an
aggregate window of 4179 packets. Given a round-trip time of 100 ms
and a packet size of 1500 bytes, this would occur with an available
bandwidth for the two competing flows of 501 Mbps. Thus, because the
Linear HighSpeed TCP is more aggressive than the HighSpeed TCP
proposed above, it also is less fair when competing with Standard TCP
in a high-bandwidth environment.

9. Tradeoffs for Choosing Congestion Control Parameters

A range of metrics can be used for evaluating choices for congestion
control parameters for HighSpeed TCP. My assumption in this section
is that for a response function of the form $w = c/p^d$, for constant c
and exponent d, the only response functions that would be considered
are response functions with $1/2 <= d <= 1$. The two ends of this
spectrum are represented by current TCP, with $d = 1/2$, and by the
linear response function described in Section 8 above, with $d = 1$.
HighSpeed TCP lies somewhere in the middle of the spectrum, with $d =
0.835$.

Response functions with exponents less than 1/2 can be eliminated
from consideration because they would be even worse than standard TCP
in accommodating connections with high congestion windows.

9.1. The Number of Round-Trip Times between Loss Events

   Response functions with exponents greater than 1 can be eliminated
   from consideration because for these response functions, the number
   of round-trip times between loss events decreases as congestion
   decreases. For a response function of w = c/p^d, with one loss event
   or congestion event every 1/p packets, the number of round-trip times
   between loss events is w^((1/d)-1)/c^(1/d). Thus, for standard TCP
   the number of round-trip times between loss events is linear in w.
   In contrast, one attraction of the linear response function, as
   described in Section 8 above, is that it is scale-invariant, in terms
   of a fixed increase in the congestion window per acknowledgement, and
   a fixed number of round-trip times between loss events.

   However, for a response function with d > 1, the number of round-
   trip times between loss events would be proportional to w^((1/d)-1),
   for a negative exponent ((1/d)-1), setting smaller as w increases.
   This would seem undesirable.

9.2. The Number of Packet Drops per Loss Event, with Drop-Tail

   A TCP connection increases its sending rate by a(w) packets per
   round-trip time, and in a Drop-Tail environment, this is likely to
   result in a(w) dropped packets during a single loss event. One
   attraction of standard TCP is that it has a fixed increase per
   round-trip time of one packet, minimizing the number of packets that
   would be dropped in a Drop-Tail environment. For an environment with
   some form of Active Queue Management, and in particular for an
   environment that uses ECN, the number of packets dropped in a single
   congestion event would not be a problem. However, even in these
   environments, larger increases in the sending rate per round-trip
   time result in larger stresses on the ability of the queues in the
   router to absorb the fluctuations.

   HighSpeed TCP plays a middle ground between the metrics of a moderate
   number of round-trip times between loss events, and a moderate
   increase in the sending rate per round-trip time. As shown in
   Appendix B, for a congestion window of 83,000 packets, HighSpeed TCP
   increases its sending rate by 70 packets per round-trip time,
   resulting in at most 70 packet drops when the buffer overflows in a
   Drop-Tail environment. This increased aggressiveness is the price
   paid by HighSpeed TCP for its increased scalability. A large number
   of packets dropped per congestion event could result in synchronized
   drops from multiple flows, with a possible loss of throughput as a
   result.

Scalable TCP has an increase a(w) of 0.005 w packets per round-trip
time. For a congestion window of 83,000 packets, this gives an
increase of 415 packets per round-trip time, resulting in roughly 415
packet drops per congestion event in a Drop-Tail environment.

Thus, HighSpeed TCP and its variants place increased demands on queue
management in routers, relative to Standard TCP. (This is rather
similar to the increased demands on queue management that would
result from using N parallel TCP connections instead of a single
Standard TCP connection.)

10. Related Issues

10.1. Slow-Start

A companion internet-draft on "Limited Slow-Start for TCP with Large
Congestion Windows" [F02b] proposes a modification to TCP's slow-
start procedure that can significantly improve the performance of TCP
connections slow-starting up to large congestion windows. For TCP
connections that are able to use congestion windows of thousands (or
tens of thousands) of MSS-sized segments (for MSS the sender's
MAXIMUM SEGMENT SIZE), the current slow-start procedure can result in
increasing the congestion window by thousands of segments in a single
round-trip time. Such an increase can easily result in thousands of
packets being dropped in one round-trip time. This is often
counter-productive for the TCP flow itself, and is also hard on the
rest of the traffic sharing the congested link.

[F02b] proposes Limited Slow-Start, limiting the number of segments
by which the congestion window is increased for one window of data
during slow-start, in order to improve performance for TCP
connections with large congestion windows. We have separated out
Limited Slow-Start to a separate draft because it can be used both
with Standard or with HighSpeed TCP.

Limited Slow-Start is illustrated in the NS simulator, for snapshots
after May 1, 2002, in the tests "./test-all-tcpHighspeed tcp1A" and
"./test-all-tcpHighspeed tcpHighspeed1" in the subdirectory
"tcl/lib".

In order for best-effort flows to safely start-up faster than slow-
start, e.g., in future high-bandwidth networks, we believe that it
would be necessary for the flow to have explicit feedback from the
routers along the path. There are a number of proposals for this,
ranging from a minimal proposal for an IP option that allows TCP SYN
packets to collect information from routers along the path about the
allowed initial sending rate [J02], to proposals with more power that
require more fine-tuned and continuous feedback from routers. These

proposals are all somewhat longer-term proposals than the HighSpeed
TCP proposal in this document, requiring longer lead times and more
coordination for deployment, and will be discussed in later
documents.

10.2. Limiting burstiness on short time scales

Because the congestion window achieved by a HighSpeed TCP connection
could be quite large, there is a possibility for the sender to send a
large burst of packets in response to a single acknowledgement. This
could happen, for example, when there is congestion or reordering on
the reverse path, and the sender receives an acknowledgement
acknowledging hundreds or thousands of new packets. Such a burst
would also result if the application was idle for a short period of
time less than a round-trip time, and then suddenly had lots of data
available to send. In this case, it would be useful for the
HighSpeed TCP connection to have some method for limiting bursts.

In this document, we do not specify TCP mechanisms for reducing the
short-term burstiness. One possible mechanism is to use some form of
rate-based pacing, and another possibility is to use maxburst, which
limits the number of packets that are sent in response to a single
acknowledgement. We would caution, however, against a permanent
reduction in the congestion window as a mechanism for limiting
short-term bursts. Such a mechanism has been deployed in some TCP
stacks, and our view would be that using permanent reductions of the
congestion window to reduce transient bursts would be a bad idea
[Fl03].

10.3. Other limitations on window size

The TCP header uses a 16-bit field to report the receive window size
to the sender. Unmodified, this allows a window size of at most
2**16 = 65K bytes. With window scaling, the maximum window size is
2**30 = 1073M bytes [RFC 1323]. Given 1500-byte packets, this allows
a window of up to 715,000 packets.

10.4. Implementation issues

One implementation issue that has been raised with HighSpeed TCP is
that with congestion windows of 4MB or more, the handling of
successive SACK packets after a packet is dropped becomes very time-
consuming at the TCP sender [S03]. Tom Kelly's Scalable TCP includes
a "SACK Fast Path" patch that addresses this problem.

The issues addressed in the Web100 project, the Net100 project, and
related projects about the tuning necessary to achieve high bandwidth
data rates with TCP apply to HighSpeed TCP as well [Net100, Web100].

11. Deployment issues

11.1. Deployment issues of HighSpeed TCP

   We do not claim that the HighSpeed TCP modification to TCP described
   in this paper is an optimal transport protocol for high-bandwidth
   environments. Based on our experiences with HighSpeed TCP in the NS
   simulator [NS], on simulation studies [SA03], and on experimental
   reports [ABLLS03,D02,CC03,F03], we believe that HighSpeed TCP
   improves the performance of TCP in high-bandwidth environments, and
   we are documenting it for the benefit of the IETF community. We
   encourage the use of HighSpeed TCP, and of its underlying response
   function, and we further encourage feedback about operational
   experiences with this or related modifications.

   We note that in environments typical of much of the current Internet,
   HighSpeed TCP behaves exactly as does Standard TCP today. This is
   the case any time the congestion window is less than 38 segments.

Bandwidth	Avg Cwnd w (pkts)	Increase a(w)	Decrease b(w)
1.5 Mbps	12.5	1	0.50
10 Mbps	83	1	0.50
100 Mbps	833	6	0.35
1 Gbps	8333	26	0.22
10 Gbps	83333	70	0.10

   Table 9: Performance of a HighSpeed TCP connection

   To help calibrate, Table 9 considers a TCP connection with 1500-byte
   packets, an RTT of 100 ms (including average queueing delay), and no
   competing traffic, and shows the average congestion window if that
   TCP connection had a pipe all to itself and fully used the link
   bandwidth, for a range of bandwidths for the pipe. This assumes that
   the TCP connection would use Table 12 in determining its increase and
   decrease parameters. The first column of Table 9 gives the
   bandwidth, and the second column gives the average congestion window
   w needed to utilize that bandwidth. The third column shows the
   increase a(w) in segments per RTT for window w. The fourth column
   shows the decrease b(w) for that window w (where the TCP sender
   decreases the congestion window from w to w(1-b(w)) segments after a
   loss event). When a loss occurs we note that the actual congestion
   window is likely to be greater than the average congestion window w
   in column 2, so the decrease parameter used could be slightly smaller
   than the one given in column 4 of Table 9.

   Table 9 shows that a HighSpeed TCP over a 10 Mbps link behaves
   exactly the same as a Standard TCP connection, even in the absence of

competing traffic. One can think of the congestion window staying
generally in the range of 55 to 110 segments, with the HighSpeed TCP
behavior being exactly the same as the behavior of Standard TCP. (If
the congestion window is ever 128 segments or more, then the
HighSpeed TCP increases by two segments per RTT instead of by one,
and uses a decrease parameter of 0.44 instead of 0.50.)

Table 9 shows that for a HighSpeed TCP connection over a 100 Mbps
link, with no competing traffic, HighSpeed TCP behaves roughly as
aggressively as six parallel TCP connections, increasing its
congestion window by roughly six segments per round-trip time, and
with a decrease parameter of roughly 1/3 (corresponding to decreasing
down to 2/3-rds of its old congestion window, rather than to half, in
response to a loss event).

For a Standard TCP connection in this environment, the congestion
window could be thought of as generally varying in the range of 550
to 1100 segments, with an average packet drop rate of $2.2 * 10^{-6}$
(corresponding to a bit error rate of $1.8 * 10^{-10}$), or equivalently,
roughly 55 seconds between congestion events. While a Standard TCP
connection could sustain such a low packet drop rate in a carefully
controlled environment with minimal competing traffic, we would
contend that in an uncontrolled best-effort environment with even a
small amount of competing traffic, the occasional congestion events
from smaller competing flows could easily be sufficient to prevent a
Standard TCP flow with no lower-speed bottlenecks from fully
utilizing the available bandwidth of the underutilized 100 Mbps link.

That is, we would contend that in the environment of 100 Mbps links
with a significant amount of available bandwidth, Standard TCP would
sometimes be unable to fully utilize the link bandwidth, and that
HighSpeed TCP would be an improvement in this regard. We would
further contend that in this environment, the behavior of HighSpeed
TCP is sufficiently close to that of Standard TCP that HighSpeed TCP
would be safe to deploy in the current Internet. We note that
HighSpeed TCP can only use high congestion windows if allowed by the
receiver's advertised window size. As a result, even if HighSpeed
TCP was ubiquitously deployed in the Internet, the impact would be
limited to those TCP connections with an advertised window from the
receiver of 118 MSS or larger.

We do not believe that the deployment of HighSpeed TCP would serve as
a block to the possible deployment of alternate experimental
protocols for high-speed congestion control, such as Scalable TCP,
XCP [KHR02], or FAST TCP [JWL03]. In particular, we don't expect
HighSpeed TCP to interact any more poorly with alternative
experimental proposals than would the N parallel TCP connections
commonly used today in the absence of HighSpeed TCP.

11.2. Deployment issues of Scalable TCP

   We believe that Scalable TCP and HighSpeed TCP have sufficiently
   similar response functions that they could easily coexist in the
   Internet. However, we have not investigated Scalable TCP
   sufficiently to be able to claim, in this document, that Scalable TCP
   is safe for a widespread deployment in the current Internet.

Bandwidth	Avg Cwnd w (pkts)	Increase a(w)	Decrease b(w)
1.5 Mbps	12.5	1	0.50
10 Mbps	83	0.4	0.125
100 Mbps	833	4.1	0.125
1 Gbps	8333	41.6	0.125
10 Gbps	83333	416.5	0.125

   Table 10: Performance of a Scalable TCP connection.

   Table 10 shows the performance of a Scalable TCP connection with
   1500-byte packets, an RTT of 100 ms (including average queueing
   delay), and no competing traffic. The TCP connection is assumed to
   use delayed acknowledgements. The first column of Table 10 gives the
   bandwidth, the second column gives the average congestion window
   needed to utilize that bandwidth, and the third and fourth columns
   give the increase and decrease parameters.

   Note that even in an environment with a 10 Mbps link, Scalable TCP's
   behavior is considerably different from that of Standard TCP. The
   increase parameter is smaller than that of Standard TCP, and the
   decrease is smaller also, 1/8-th instead of 1/2. That is, for 10
   Mbps links, Scalable TCP increases less aggressively than Standard
   TCP or HighSpeed TCP, but decreases less aggressively as well.

   In an environment with a 100 Mbps link, Scalable TCP has an increase
   parameter of roughly four segments per round-trip time, with the same
   decrease parameter of 1/8-th. A comparison of Tables 9 and 10 shows
   that for this scenario of 100 Mbps links, HighSpeed TCP increases
   more aggressively than Scalable TCP.

   Next we consider the relative fairness between Standard TCP,
   HighSpeed TCP and Scalable TCP. The relative fairness between
   HighSpeed TCP and Standard TCP was shown in Table 5 earlier in this
   document, and the relative fairness between Scalable TCP and Standard
   TCP was shown in Table 8. Following the approach in Section 6, for a
   given packet drop rate p, for $p < 10^{-3}$, we can estimate the relative
   fairness between Scalable and HighSpeed TCP as
   W_Scalable/W_HighSpeed. This relative fairness is shown in Table 11
   below. The bandwidth in the last column of Table 11 is the aggregate

bandwidth of the two competing flows given 100 ms round-trip times
and 1500-byte packets.

Packet Drop Rate P	Fairness	Aggregate Window	Bandwidth
10^{-2}	1.0	24	2.8 Mbps
10^{-3}	1.0	76	9.1 Mbps
10^{-4}	1.4	643	77.1 Mbps
10^{-5}	2.1	5595	671.4 Mbps
10^{-6}	3.1	50279	6.0 Gbps
10^{-7}	4.5	463981	55.7 Gbps

Table 11: Relative Fairness between the Scalable and HighSpeed
Response Functions.

The second row of Table 11 shows that for a Scalable TCP and a
HighSpeed TCP flow competing in an environment with 100 ms RTTs and a
10 Mbps pipe, the two flows would receive essentially the same
bandwidth. The next row shows that for a Scalable TCP and a
HighSpeed TCP flow competing in an environment with 100 ms RTTs and a
100 Mbps pipe, the Scalable TCP flow would receive roughly 50% more
bandwidth than would HighSpeed TCP. Table 11 shows the relative
fairness in higher-bandwidth environments as well. This relative
fairness seems sufficient that there should be no problems with
Scalable TCP and HighSpeed TCP coexisting in the same environment as
Experimental variants of TCP.

We note that one question that requires more investigation with
Scalable TCP is that of convergence to fairness in environments with
Drop-Tail queue management.

12. Related Work in HighSpeed TCP

   HighSpeed TCP has been separately investigated in simulations by
   Sylvia Ratnasamy and by Evandro de Souza [SA03]. The simulations in
   [SA03] verify the fairness properties of HighSpeed TCP when sharing a
   link with Standard TCP.

   These simulations explore the relative fairness of HighSpeed TCP
   flows when competing with Standard TCP. The simulation environment
   includes background forward and reverse-path TCP traffic limited by
   the TCP receive window, along with a small amount of forward and
   reverse-path traffic from the web traffic generator. Most of the
   simulations so far explore performance on a simple dumbbell topology
   with a 1 Gbps link with a propagation delay of 50 ms. Simulations
   have been run with Adaptive RED and with DropTail queue management.

The simulations in [SA03] explore performance with a varying number
of competing flows, with the competing traffic being all standard
TCP; all HighSpeed TCP; or a mix of standard and HighSpeed TCP. For
the simulations in [SA03] with RED queue management, the relative
fairness between standard and HighSpeed TCP is consistent with the
relative fairness predicted in Table 5. For the simulations with
Drop Tail queues, the relative fairness is more skewed, with the
HighSpeed TCP flows receiving an even larger share of the link
bandwidth. This is not surprising; with Active Queue Management at
the congested link, the fraction of packet drops received by each
flow should be roughly proportional to that flow's share of the link
bandwidth, while this property no longer holds with Drop Tail queue
management. We also note that relative fairness in simulations with
Drop Tail queue management can sometimes depend on small details of
the simulation scenario, and that Drop Tail simulations need special
care to avoid phase effects [F92].

[SA03] explores the bandwidth 'stolen' by HighSpeed TCP from standard
TCP by exploring the fraction of the link bandwidth N standard TCP
flows receive when competing against N other standard TCP flows, and
comparing this to the fraction of the link bandwidth the N standard
TCP flows receive when competing against N HighSpeed TCP flows. For
the 1 Gbps simulation scenarios dominated by long-lived traffic, a
small number of standard TCP flows are able to achieve high link
utilization, and the HighSpeed TCP flows can be viewed as stealing
bandwidth from the competing standard TCP flows, as predicted in
Section 6 on the Fairness Implications of the HighSpeed Response
Function. However, [SA03] shows that when even a small fraction of
the link bandwidth is used by more bursty, short TCP connections, the
standard TCP flows are unable to achieve high link utilization, and
the HighSpeed TCP flows in this case are not 'stealing' bandwidth
from the standard TCP flows, but instead are using bandwidth that
otherwise would not be utilized.

The conclusions of [SA03] are that "HighSpeed TCP behaved as forseen
by its response function, and appears to be a real and viable option
for use on high-speed wide area TCP connections."

Future work that could be explored in more detail includes
convergence times after new flows start-up; recovery time after a
transient outage; the response to sudden severe congestion, and
investigations of the potential for oscillations. We invite
contributions from others in this work.

13. Relationship to other Work

   Our assumption is that HighSpeed TCP will be used with the TCP SACK
   option, and also with the increased Initial Window of three or four
   segments, as allowed by [RFC3390]. For paths that have substantial
   reordering, TCP performance would be greatly improved by some of the
   mechanisms still in the research stages for robust performance in the
   presence of reordered packets.

   Our view is that HighSpeed TCP is largely orthogonal to proposals for
   higher PMTU (Path MTU) values [M02]. Unlike changes to the PMTU,
   HighSpeed TCP does not require any changes in the network or at the
   TCP receiver, and works well in the current Internet. Our assumption
   is that HighSpeed TCP would be useful even with larger values for the
   PMTU. Unlike the current congestion window, the PMTU gives no
   information about the bandwidth-delay product available to that
   particular flow.

   A related approach is that of a virtual MTU, where the actual MTU of
   the path might be limited [VMSS,S02]. The virtual MTU approach has
   not been fully investigated, and we do not explore the virtual MTU
   approach further in this document.

14. Conclusions

   This document has proposed HighSpeed TCP, a modification to TCP's
   congestion control mechanism for use with TCP connections with large
   congestion windows. We have explored this proposal in simulations,
   and others have explored HighSpeed TCP with experiments, and we
   believe HighSpeed TCP to be safe to deploy on the current Internet.
   We would welcome additional analysis, simulations, and particularly,
   experimentation. More information on simulations and experiments is
   available from the HighSpeed TCP Web Page [HSTCP]. There are several
   independent implementations of HighSpeed TCP [D02,F03] and of
   Scalable TCP [K03] for further investigation.

15. Acknowledgements

   The HighSpeed TCP proposal is from joint work with Sylvia Ratnasamy
   and Scott Shenker (and was initiated by Scott Shenker). Additional
   investigations of HighSpeed TCP were joint work with Evandro de Souza
   and Deb Agarwal. We thank Tom Dunigan for the implementation in the
   Linux 2.4.16 Web100 kernel, and for resulting experimentation with
   HighSpeed TCP. We are grateful to the End-to-End Research Group, the
   members of the Transport Area Working Group, and to members of the
   IPAM program in Large Scale Communication Networks for feedback. We
   thank Glenn Vinnicombe for framing the Linear response function in
   the parameters of HighSpeed TCP. We are also grateful for

contributions and feedback from the following individuals: Les
Cottrell, Mitchell Erblich, Jeffrey Hsu, Tom Kelly, Chuck Jackson,
Matt Mathis, Jitendra Padhye, Andrew Reiter, Stanislav Shalunov, Alex
Solan, Paul Sutter, Brian Tierney, Joe Touch.

16. Normative References

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

17. Informative References

   [ABLLS03] A. Antony, J. Blom, C. de Laat, J. Lee, and W. Sjouw,
              "Microscopic Examination of TCP Flows over Transatlantic
              Links", iGrid2002 special issue, Future Generation
              Computer Systems, volume 19 issue 6 (2003), URL
              "http://www.science.uva.nl/~delaat/techrep-2003-2-
              tcp.pdf".

   [BBFS01]   Deepak Bansal, Hari Balakrishnan, Sally Floyd, and Scott
              Shenker, "Dynamic Behavior of Slowly-Responsive Congestion
              Control Algorithms", SIGCOMM 2001, August 2001.

   [CC03]   Fabrizio Coccetti and Les Cottrell, "TCP Stack
              Measurements on Lightly Loaded Testbeds", 2003. URL
              "http://www-iepm.slac.stanford.edu/monitoring/bulk/fast/".

   [CJ89]   D. Chiu and R. Jain, "Analysis of the Increase and
              Decrease Algorithms for Congestion Avoidance in Computer
              Networks", Computer Networks and ISDN Systems, Vol. 17,
              pp. 1-14, 1989.

   [CO98]   J. Crowcroft and P. Oechslin, "Differentiated End-to-end
              Services using a Weighted Proportional Fair Share TCP",
              Computer Communication Review, 28(3):53--69, 1998.

   [D02]   Tom Dunigan, "Floyd's TCP slow-start and AIMD mods", URL
              "http://www.csm.ornl.gov/~dunigan/net100/floyd.html".

   [F03]   Gareth Fairey, "High-Speed TCP", 2003. URL
              "http://www.hep.man.ac.uk/u/garethf/hstcp/".

   [F92]   S. Floyd and V. Jacobson, "On Traffic Phase Effects in
              Packet-Switched Gateways, Internetworking: Research and
              Experience", V.3 N.3, September 1992, p.115-156. URL
              "http://www.icir.org/floyd/papers.html".

   [Fl03]   Sally Floyd, "Re: [Tsvwg] taking NewReno (RFC 2582) to
              Proposed Standard", Email to the tsvwg mailing list, May
              14, 2003.

   URLs   "http://www1.ietf.org/mail-archive/working-
              groups/tsvwg/current/msg04086.html" and
              "http://www1.ietf.org/mail-archive/working-
              groups/tsvwg/current/msg04087.html".

   [FF98]   Floyd, S., and Fall, K., "Promoting the Use of End-to-End
              Congestion Control in the Internet", IEEE/ACM Transactions
              on Networking, August 1999.

   [FRS02]   Sally Floyd, Sylvia Ratnasamy, and Scott Shenker,
              "Modifying TCP's Congestion Control for High Speeds", May
              2002. URL "http://www.icir.org/floyd/notes.html".

   [GRK99]   Panos Gevros, Fulvio Risso and Peter Kirstein, "Analysis
              of a Method for Differential TCP Service". In Proceedings
              of the IEEE GLOBECOM'99, Symposium on Global Internet ,
              December 1999, Rio de Janeiro, Brazil.

   [GV02]   S. Gorinsky and H. Vin, "Extended Analysis of Binary
              Adjustment Algorithms", Technical Report TR2002-39,
              Department of Computer Sciences, The University of Texas
              at Austin, August 2002. URL
              "http://www.cs.utexas.edu/users/gorinsky/pubs.html".

   [HSTCP]   HighSpeed TCP Web Page, URL
              "http://www.icir.org/floyd/hstcp.html".

   [J02]   Amit Jain and Sally Floyd, "Quick-Start for TCP and IP",
              Work in Progress, 2002.

   [JWL03]   Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP for
              High-speed Long-distance Networks", Work in Progress, June
              2003.

   [K03]   Tom Kelly, "Scalable TCP: Improving Performance in
              HighSpeed Wide Area Networks", February 2003. URL
              "http://www-lce.eng.cam.ac.uk/˜ctk21/scalable/".

   [KHR02]   Dina Katabi, Mark Handley, and Charlie Rohrs, "Congestion
              Control for High Bandwidth-Delay Product Networks",
              SIGCOMM 2002.

   [M02]   Matt Mathis, "Raising the Internet MTU", Web Page, URL
              "http://www.psc.edu/˜mathis/MTU/".

   [Net100]   The DOE/MICS Net100 project. URL
               "http://www.csm.ornl.gov/~dunigan/net100/".

   [NS]   The NS Simulator, "http://www.isi.edu/nsnam/ns/".

   [RFC 1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions
               for High Performance", RFC 1323, May 1992.

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

   [RFC3448]   Handley, M., Padhye, J., Floyd, S. and J. Widmer, "TCP
               Friendly Rate Control (TFRC): Protocol Specification", RFC
               3448, January 2003.

   [SA03]   Souza, E. and D.A., Agarwal, "A HighSpeed TCP Study:
               Characteristics and Deployment Issues", LBNL Technical
               Report LBNL-53215. URL
               "http://www.icir.org/floyd/hstcp.html".

   [S02]   Stanislav Shalunov, "TCP Armonk", Work in Progress, 2002,
               URL "http://www.internet2.edu/~shalunov/tcpar/".

   [S03]   Alex Solan, private communication, 2003.

   [VMSS]   "Web100 at ORNL", Web Page,
               "http://www.csm.ornl.gov/~dunigan/netperf/web100.html".

   [Web100]   The Web100 project. URL "http://www.web100.org/".

18. Security Considerations

   This proposal makes no changes to the underlying security of TCP.

19. IANA Considerations

   There are no IANA considerations regarding this document.

A. TCP's Loss Event Rate in Steady-State

   This section gives the number of round-trip times between congestion
   events for a TCP flow with D-byte packets, for D=1500, as a function
   of the connection's average throughput B in bps. To achieve this
   average throughput B, a TCP connection with round-trip time R in
   seconds requires an average congestion window w of BR/(8D) segments.

   In steady-state, TCP's average congestion window w is roughly
   $1.2/\sqrt{p}$ segments. This is equivalent to a lost event at most
   once every 1/p packets, or at most once every 1/(pw) = w/1.5 round-
   trip times. Substituting for w, this is a loss event at most every
   (BR)/12D)round-trip times.

   An an example, for R = 0.1 seconds and D = 1500 bytes, this gives
   B/180000 round-trip times between loss events.

B. A table for a(w) and b(w).

    This section gives a table for the increase and decrease parameters
    a(w) and b(w) for HighSpeed TCP, for the default values of Low_Window
    = 38, High_Window = 83000, High_P = 10^-7, and High_Decrease = 0.1.

w	a(w)	b(w)
38	1	0.50
118	2	0.44
221	3	0.41
347	4	0.38
495	5	0.37
663	6	0.35
851	7	0.34
1058	8	0.33
1284	9	0.32
1529	10	0.31
1793	11	0.30
2076	12	0.29
2378	13	0.28
2699	14	0.28
3039	15	0.27
3399	16	0.27
3778	17	0.26
4177	18	0.26
4596	19	0.25
5036	20	0.25
5497	21	0.24
5979	22	0.24
6483	23	0.23
7009	24	0.23
7558	25	0.22
8130	26	0.22
8726	27	0.22
9346	28	0.21
9991	29	0.21
10661	30	0.21
11358	31	0.20
12082	32	0.20
12834	33	0.20
13614	34	0.19
14424	35	0.19
15265	36	0.19
16137	37	0.19
17042	38	0.18
17981	39	0.18
18955	40	0.18

```
19965   41  0.17
21013   42  0.17
22101   43  0.17
23230   44  0.17
24402   45  0.16
25618   46  0.16
26881   47  0.16
28193   48  0.16
29557   49  0.15
30975   50  0.15
32450   51  0.15
33986   52  0.15
35586   53  0.14
37253   54  0.14
38992   55  0.14
40808   56  0.14
42707   57  0.13
44694   58  0.13
46776   59  0.13
48961   60  0.13
51258   61  0.13
53677   62  0.12
56230   63  0.12
58932   64  0.12
61799   65  0.12
64851   66  0.11
68113   67  0.11
71617   68  0.11
75401   69  0.10
79517   70  0.10
84035   71  0.10
89053   72  0.10
94717   73  0.09
```

   Table 12: Parameters for HighSpeed TCP.

This table was computed with the following Perl program:

```
$top = 100000;
$num = 38;
if ($num == 38) {
  print "     w  a(w)  b(w)\n";
  print "  ----  ----  ----\n";
  print "    38    1  0.50\n";
  $oldb = 0.50;
  $olda = 1;
}
while ($num < $top) {
  $bw = (0.1 -0.5)*(log($num)-log(38))/(log(83000)-log(38))+0.5;
  $aw = ($num**2*2.0*$bw) / ((2.0-$bw)*$num**1.2*12.8);
  if ($aw > $olda + 1) {
     printf "%6d %5d  %3.2f0, $num, $aw, $bw;
     $olda = $aw;
  }
  $num ++;
}
```

Table 13: Perl Program for computing parameters for HighSpeed TCP.

C. Exploring the time to converge to fairness.

   This section gives the Perl program used to compute the congestion
   window growth during congestion avoidance.

```
    $top = 2001;
    $hswin = 1;
    $regwin = 1;
    $rtt = 1;
    $lastrtt = 0;
    $rttstep = 100;
    if ($hswin == 1) {
      print "  RTT  HS_Window Standard_TCP_Window0;
      print "  ---  --------- ------------------0;
    }
    while ($rtt < $top) {
      $bw = (0.1 -0.5)*(log($hswin)-log(38))/(log(83000)-log(38))+0.5;
      $aw = ($hswin**2*2.0*$bw) / ((2.0-$bw)*$hswin**1.2*12.8);
      if ($aw < 1) {
          $aw = 1;
      }
      if ($rtt >= $lastrtt + $rttstep) {
        printf "%5d %9d %10d0, $rtt, $hswin, $regwin;
        $lastrtt = $rtt;
      }
      $hswin += $aw;
      $regwin += 1;
      $rtt ++;
     }
```

   Table 14: Perl Program for computing the window in congestion
   avoidance.

Author's Address

   Sally Floyd
   ICIR (ICSI Center for Internet Research)

   Phone: +1 (510) 666-2989
   EMail: floyd@acm.org
   URL: http://www.icir.org/floyd/

Full Copyright Statement

Acknowledgement