Internet Draft
draft-hkchu-tcpm-initcwnd-01.txt
Intended status: Standard
Updates: 3390, 5681
Creation date: July 12, 2010
Expiration date: January 2011

J. Chu
N. Dukkipati
Y. Cheng
M. Mathis
Google, Inc.

Increasing TCP's Initial Window

Status of this Memo

Copyright Notice

Abstract

This document proposes an increase in the permitted TCP initial window (IW) from between 2 and 4 segments, as specified in RFC 3390, to 10 segments. It discusses the motivation behind the increase, the advantages and disadvantages of the higher initial window, and presents results from several large scale experiments showing that the higher initial window improves the overall performance of many web services without risking congestion collapse. Finally, it outlines a list of concerns to be addressed in future tests.

Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Editor's Note

This draft aims at updating RFC 3390, thus it follows RFC 3390's layout closely. Much of the analysis from RFC 3390 remains valid. Some non-critical details are intentionally excluded from this draft. The intent is to have the draft published to solicit feedbacks early. All the excluded pieces will be supplied in later revisions.

The choice of 10 for initial window may not be the "optimal" one. Our most recent tests gave better performance with IW=16 compared to IW=10, while still showing little negative impact. We are still in the process of completing the latest large scale tests and analysis for IW=16, which might be a better, more future proofing choice.

1. Introduction

TCP congestion window was introduced as part of the congestion control algorithm by Van Jacobson in 1988 [Jac88]. The initial value of one segment was used as the starting point for newly established connections to probe the available bandwidth on the network.

The default value was increased to roughly 4KB more than a decade ago [RFC2414]. Since then, the Internet has continued to grow, both in speed and penetration [AKAM10]. Today's Internet is dominated by web traffic running on top of short-lived TCP connections [IOR2009]. The relatively small initial window has become a limiting factor for the performance of many web applications.

This document proposes an optional standard to allow TCP's initial window to start at 10 segments or roughly 15KB, updating RFC 3390 [RFC3390]. It discusses the motivation, the advantages and

disadvantages of the higher initial window, and includes test results from several large scale experiments showing improved latency across the board for a variety of BW, RTT, and BDP classes.

It also discusses potential negative impacts and suggests mitigation. A minor change to RFC 3390 and RFC 5681 [RFC5681] is proposed on resetting the initial window when the SYN or SYN/ACK is lost.

The document closes with a discussion on remaining concerns, and future tests to further validate the higher initial window.

2. TCP Modification

This document proposes an increase in the permitted upper bound for TCP's initial window (IW) to 10 segments. This increase is optional: a TCP MAY start with a larger initial window up to 10 segments.

This upper bound for the initial window size represents a change from RFC 3390 [RFC3390], which specified that the congestion window be initialized between 2 and 4 segments depending on the MSS.

This change applies to the initial window of the connection in the first round trip time (RTT) of data transmission following the TCP three-way handshake. Neither the SYN/ACK nor its acknowledgment (ACK) in the three-way handshake should increase the initial window size beyond 10 segments.

Furthermore, RFC 3390 and RFC 5681 [RFC5681] state that

> "If the SYN or SYN/ACK is lost, the initial window used by a sender after a correctly transmitted SYN MUST be one segment consisting of MSS bytes."

The proposed change to reduce the default RTO to 1 second [PAC10] increases the chance for spurious SYN or SYN/ACK retransmission, thus unnecessarily penalizing connections with RTT > 1 second if their initial window is reduced to 1 segment. For this reason, it is RECOMMENDED that implementations refrain from resetting the initial window to 1 segment, unless either there have been multiple SYN or SYN/ACK retransmissions, or true loss detection has been made.

TCP implementations use slow start in as many as three different ways: (1) to start a new connection (the initial window); (2) to restart transmission after a long idle period (the restart window); and (3) to restart transmission after a retransmit timeout (the loss window). The change specified in this document affects the value of the initial window. Optionally, a TCP MAY set the restart window to the minimum of the value used for the initial window and the current

value of cwnd (in other words, using a larger value for the restart window should never increase the size of cwnd). These changes do NOT change the loss window, which must remain 1 segment of MSS bytes (to permit the lowest possible window size in the case of severe congestion).

Furthermore, to limit any negative effect that a larger initial window may have on links with limited bandwidth or buffer space, implementations SHOULD fall back to RFC 3390 for the restart window (RW), if any packet loss is detected during either the initial window, or a restart window, when more than 4KB of data is sent.

3. Motivation

The global Internet has continued to grow, both in speed and penetration. According to the latest report from Akamai [AKAM10], the global broadband (> 2Mbps) adoption has surpassed 50%, propelling the average connection speed to reach 1.7Mbps, while the narrowband (< 256Kbps) usage has dropped to 5%. In contrast, TCP's initial window has remained 4KB for a decade, corresponding to a bandwidth utilization of less than 200Kbps per connection, assuming an RTT of 200ms.

A large proportion of flows on the Internet are short web transactions over TCP, and complete before exiting TCP slow start. Speeding up the TCP flow startup phase, including circumventing the initial window limit, has been an area of active research [PWSB09, Sch08]. Numerous proposals exist [LAJW07, RFC4782, PRAKS02, PK98]. Some require router support [RFC4782, PK98], hence are not practical for the public Internet. Others suggested bold, but often radical ideas, likely requiring more years of research before standardization and deployment.

In the mean time, applications have responded to TCP's "slow" start. Web sites use multiple sub-domains [Bel10] to circumvent HTTP 1.1 regulation on two connections per physical host [RFC2616]. As of today, major web browsers open multiple connections to the same site (up to six connections per domain [Ste08] and the number is growing). This trend is to remedy HTTP serialized download to achieve parallelism and higher performance. But it also implies today most access links are severely under-utilized, hence having multiple TCP connections improves performance most of the time. While raising the initial congestion window may cause congestion for certain users using these browsers, we argue that the browsers and other application need to respect HTTP 1.1 regulation and stop increasing number of simultaneous TCP connections. We believe a modest increase of the initial window will help to stop this trend, and provide the best interim solution to improve overall user performance, and reduce

the server, client, and network load.

Note that persistent connections and pipelining are designed to address some of the issues with HTTP above [RFC2616]. Their presence does not diminish the need for a larger initial window, as the first data chunk to respond is often the largest, and will easily hit the initial window limit. Our test data confirm significant latency reduction with the large initial window even with these two HTTP features ([Duk10]).

Also note that packet pacing has been suggested as an effective mechanism to avoid large bursts and their associated damage [VH97]. We do not require pacing in our proposal due to our strong preference for a simple solution. We suspect for packet bursts of a moderate size, packet pacing will not be necessary. This seems to be confirmed by our test results.

More discussion of the increase in initial window, including the choice of 10 segments can be found in [Duk10].

4. Implementation Issues

[Need to decide if a different formula is needed for PMTU != 1500.]

HTTP 1.1 specification allows only two simultaneous connections per domain, while web browsers open more simultaneous TCP connections [Ste08], partly to circumvent the small initial window in order to speed up the loading of web pages as described above.

When web browsers open simultaneous TCP connections to the same destination, they are working against TCP's congestion control mechanisms [FF99]. Combining this behavior with larger initial windows further increases the burstiness and unfairness to other traffic in the network. A larger initial window will incent applications to use fewer concurrent TCP connections.

Some implementations advertise small initial receive window (Table 2 in [Duk10]), effectively limiting how much window a remote host may use. In order to realize the full benefit of the large initial window, implementations are encouraged to advertise an initial receive window of at least 10 segments, except for the circumstances where a larger initial window is deemed harmful. (See the Mitigation section below.)

5. Advantages of Larger Initial Windows

1. Reducing Latency

An increase of the initial window from 3 segments to 10 segments reduces the total transfer time for data sets greater than 4KB by up to 4 round trips.

The table below compares the number of round trips between IW=3 and IW=10 for different transfer sizes, assuming infinite bandwidth, no packet loss, and the standard delayed acks with large delay-ack timer.

total segments	IW=3	IW=10
3	1	1
6	2	1
10	3	1
12	3	2
21	4	2
25	5	2
32	5	3
46	6	3
51	6	4
79	7	4
121	8	5
128	9	5

For example, with the larger initial window, a transfer of 32KB data will require only two rather than five round trips to complete.

2. Keeping up with the growth of web object size

RFC 3390 stated that the main motivation for increasing the initial window to 4KB was to speed up connections that only transmit a small amount of data, e.g., email and web. The majority of transfers back then were less than 4KB, and could be completed in a single RTT [All00].

Since RFC 3390 was published, web objects have gotten significantly larger [Chu09, RJ10]. A large percentage of web objects today no longer fit in the 4KB initial window, and will require more than one round trip to transfer. E.g., only 10% of Google's search responses can fit in 4KB, while 90% can fit in 10 segments (15KB). The average HTTP response size of gmail.com, a highly scripted web-site, is 8KB (Figure 1. in [Duk10]).

During the same period, the average web page, including all static and dynamic scripted web objects on the page, has seen

even greater growth in size [RJ10]. HTTP pipelining [RFC2616] and new web transport protocols like SPDY [SPDY] allow multiple web objects to be sent in a single transaction, potentially requiring even larger initial window in order to transfer a whole web page in one round trip.

3. Recovering faster from loss on under-utilized or wireless links

A greater-than-3-segment initial window increases the chance to recover packet loss through Fast Retransmit rather than the lengthy initial RTO [RFC5681]. This is because the fast retransmit algorithm requires three duplicate acks as an indication that a segment has been lost rather than reordered. While newer loss recovery techniques such as Limited Transmit [RFC3042] and Early Retransmit [AAABH10] have been proposed to help speeding up loss recovery from a smaller window, both algorithms can still benefit from the larger initial window because of a better chance to receive more ACKs to react upon.

6. Disadvantages of Larger Initial Windows for the Individual Connection

The larger bursts from an increase in the initial window may cause buffer overrun and packet drop in routers with small buffers, or routers experiencing congestion. This could result in unnecessary retransmit timeouts. For a large-window connection that is able to recover without a retransmit timeout, this could result in an unnecessarily-early transition from the slow-start to the congestion-avoidance phase of the window increase algorithm. [Note: knowing the large initial window may cause premature segment drop, should one make an exception for it, i.e., by allowing ssthresh to remain unchanged if loss is from an enlarged initial window?]

Premature segment drops are unlikely to occur in uncongested networks with sufficient buffering, or in moderately-congested networks where the congested router uses active queue management (such as Random Early Detection [FJ93, RFC2309, RFC3150]).

Insufficient buffering is more likely to exist in the access routers connecting slower links. A recent study of access router buffer size [DGHS07] reveals the majority of access routers provision enough buffer for 130ms or longer, sufficient to cover a burst of more than 10 packets at 1Mbps speed, but possibly not sufficient for browsers opening simultaneous connections.

Some TCP connections will receive better performance with the larger initial window even if the burstiness of the initial window results in premature segment drops. This will be true if (1) the TCP

connection recovers from the segment drop without a retransmit timeout, and (2) the TCP connection is ultimately limited to a small congestion window by either network congestion or by the receiver's advertised window.

7. Disadvantages of Larger Initial Windows for the Network

An increase in the initial window may increase congestion in a network. However, since the increase is one-time only (at the beginning of a connection), and the rest of TCP's congestion backoff mechanism remains in place, it's highly unlikely the increase will render a network in a persistent state of congestion, or even congestion collapse. This seems to have been confirmed by our large scale experiments described later.

Some of the discussions from RFC 3390 are still valid for IW=10. Moreover, it is worth noting that although TCP NewReno increases the chance of duplicate segments when trying to recover multiple packet losses from a large window [RFC3782], the wide support of TCP Selective Acknowledgment (SACK) option [RFC2018] in all major OSes today should keep the volume of duplicate segments in check.

8. Mitigation of Negative Impact

Much of the negative impact from an increase in the initial window is likely to be felt by users behind slow links with limited buffers. The negative impact can be mitigated by hosts directly connected to a low-speed link advertising a smaller initial receive window than 10 segments. This can be achieved either through manual configuration by the users, or through the host stack auto-detecting the low bandwidth links.

More suggestions to improve the end-to-end performance of slow links can be found in RFC 3150 [RFC3150].

[Note: if packet loss is detected during IW through fast retransmit, should cwnd back down to 2 rather than FlightSize / 2?]

9. Interactions with the Retransmission Timer

A large initial window increases the chance of spurious RTO on a low-bandwidth path because the packet transmission time will dominate the round-trip time. To minimize spurious retransmissions, implementations MUST follow RFC 2988 [RFC2988] to restart the retransmission timer with the current value of RTO for each ack received that acknowledges new data.

10. Experimental Results

In this section we summarize our findings from large scale Internet experiments with an initial window of 10 segments, conducted via Google's front-end infrastructure serving a diverse set of applications. We present results from two datacenters, each chosen because of the specific characteristics of subnets served: AvgDC has connection bandwidths closer to the worldwide average reported in [AKAM10], with a median connection speed of about 1.7Mbps; SlowDC has a larger proportion of traffic from slow bandwidth subnets with nearly 20% of traffic from connections below 100Kbps, and a third below 256Kbps.

Guided by measurements data, we answer two key questions: what is the latency benefit when TCP connections start with a higher initial window, and on the flip side, what is the cost?

10.1 The benefits

The average web search latency improvement over all responses in AvgDC is 11.7% (68 ms) and 8.7% (72 ms) in SlowDC. We further analyzed the data based on traffic characteristics and subnet properties such as bandwidth (BW), round-trip time (RTT), and bandwidth-delay product (BDP). The average response latency improved across the board for a variety of subnets with the largest benefits of over 20% from high RTT and high BDP networks, wherein most responses can fit within the pipe. Correspondingly, responses from low RTT paths experienced the smallest improvements of about 5%.

Contrary to what we expected, responses from low bandwidth subnets experienced the best latency improvements (between 10-20%) in the buckets 0-56Kbps and 56-256Kbps buckets. We speculate low BW networks observe improved latency for two plausible reasons: 1) fewer slow-start rounds: unlike many large BW networks, low BW subnets with dial-up modems have inherently large RTTs; and 2) faster loss recovery: an initial window larger than 3 segments increases the chances of a lost packet to be recovered through Fast Retransmit as opposed to a lengthy RTO.

Responses of different sizes benefited to varying degrees; those larger than 3 segments naturally demonstrated larger improvements, because they finished in fewer rounds in slow start as compared to the baseline. In our experiments, response sizes <= 3 segments also demonstrated small latency benefits.

To find out how individual subnets performed, we analyzed average latency at a /24 subnet level (an approximation to a user base offered similar set of services by a common ISP). We find even at the subnet granularity, latency improved at all quantiles ranging from 5-11%.

10.2 The cost

To quantify the cost of raising the initial window, we analyzed the data specifically for subnets with low bandwidth and BDP, retransmission rates for different kinds of applications, as well as latency for applications operating with multiple concurrent TCP connections. From our measurements we found no evidence of a negative latency impacts that correlate to BW or BDP alone, but in fact both kinds of subnets demonstrated latency improvements across averages and quantiles.

As expected, the retransmission rate increased modestly when operating with larger initial congestion window. The overall increase in AvgDC is 0.3% (from 1.98% to 2.29%) and in SlowDC is 0.7% (from 3.54% to 4.21%). In our investigation, with the exception of one application, the larger window resulted in a retransmission increase of < 0.5% for services in the AvgDC. The exception is the Maps application that operates with multiple concurrent TCP connections, which increased its retransmission rate by 0.9% in AvgDC and 1.85% in SlowDC (from 3.94% to 5.79%).

In our experiments, the percentage of traffic experiencing retransmissions did not increase significantly. E.g. 90% of web search and maps experienced zero retransmissions in SlowDC (percentages are higher for AvgDC); a break up of retransmissions by percentiles indicate that most increases come from portion of traffic already experiencing retransmissions in the baseline with initial window of 3 segments.

Traffic patterns from applications using multiple concurrent TCP connections all operating with a large initial window represent one of the worst case scenarios where latency can be adversely impacted due to bottleneck buffer overflow. Our investigation shows that such a traffic pattern has not been a problem in AvgDC, where all these applications, specifically maps and image thumbnails, demonstrated improved latencies varying from 2-20%. In the case of SlowDC, while these applications continued showing a latency improvement in the mean, their latencies in higher quantiles (96 and above for maps) indicated instances where latency with larger window is worse than the baseline, e.g. the 99% latency for maps has increased by 2.3% (80ms) when compared to the baseline. There is no evidence from our measurements that such a cost on latency is a result of subnet bandwidth alone. Although we have no way of knowing from our data, we conjecture that the amount of buffering at bottleneck links plays a key role in performance of these applications.

Further details on our experiments and analysis can be found in [Duk10].

11. List of Concerns and Future Tests

Although we were a little hard pressed to find negative impact from the initial window increase in our large scale tests, we don't contend our test coverage is complete. The following is an attempt to compile a list of concerns and to suggest future tests. Ultimately we would like to enlist the help from the TCP community at IETF to study and address any concern that may come up.

1. How complete are our tests in traffic pattern coverage?

   Google today offers a large portfolio of services beyond web search. The list includes Gmail, Google Maps, Photos, News, Sites, Images, Videos,..., etc. Our tests included most of Google's services, covering a wide variety of traffic sizes and patterns. One notable exception is YouTube because we don't think the large initial window will have much material impact, either positive or negative, on bulk data services.

2. Larger bursts from the increase in the initial window cause significantly more packet drops

   Let the max burst capacity of an end-to-end path be the largest burst of packets a given path can absorb before packet is dropped. To analyze the impact from the larger initial window, it helps to study the distribution of the max burst capacity of the current Internet.

   In the past similar studies were conducted by actively probing, e.g., through the TCP echo/discard ports from a large set of endhosts. However, most endhosts today are behind firewall enabled NAT boxes, making active probing infeasible.

   Our plan is to monitor TCP connections used to carry Google's bulk data services like YouTube, and infer the max burst capacity on a per-client basis from TCP internal connection parameters such as ssthresh, max cwnd, and packet drop pattern.

3. Need more thorough analysis of the impact on slow links

   Although our data showed the large initial window reduced the average latency even for the dialup link class of only 56Kbps in bandwidth, it is only prudent to perform more microscopic analysis on its effect on slow links. Moreover, data from the YouTube study above will likely be biased toward broadband users, leaving out users behind slow links.

   The narrowband classes here should include 56Kbps dialup modem,

2.5G and GPRS mobile network.

4. How will the larger initial window affect flows with initial windows 4KB or less?

   Flows with the larger initial window will likely grab more bandwidth from a bottleneck link when competing against flows with smaller initial window, at least initially. How long will this "unfairness" last? Will there be any "capture effect" where flows with larger initial window possess a disproportional share of bandwidth beyond just a few round trips?

   If there is any "unfairness" issue from flows with different initial windows, it did not show up in our large scale experiments, as the average latency for the bucket of all responses < 4KB did not seem to be affected by the presence of many other larger responses employing large initial window. As a matter of fact they seemed to benefit from the large initial window too, as shown in Figure 7 of [Duk10].

   More study can be done through simulation, similar to the set described in RFC 2415 [RFC2415].

12. Security Considerations

This document discusses the initial congestion window permitted for TCP connections. Changing this value does not raise any known new security issues with TCP.

13. Conclusion

This document suggests a change to TCP that will likely be beneficial to short-lived TCP connections and those over links with long RTTs (saving several RTTs during the initial slow-start phase). However, more tests are likely needed to fully understand its impact to the Internet. We welcome any help from the TCP community at IETF in moving this proposal forward.

14. IANA Considerations

None

Acknowledgments

Many people at Google have helped to make the set of large scale tests possible. We would especially like to acknowledge Amit Agarwal, Tom Herbert, Arvind Jain and Tiziana Refice for their major contributions.

Normative References

[PAC10]   Paxson, V., Allman, M., and J. Chu, "Computing TCP's Retransmission Timer", Internet-draft draft-paxson-tcpm-rfc2988bis-00, work in progress, February, 2010.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

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

[RFC5681] Allman, M., Paxson, V. and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

Informative References

[AAABH10] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J. and P. Hurtig, "Early Retransmit for TCP and SCTP", Internet-draft draft-ietf-tcpm-early-rexmt-04.txt, work in progress.

[AKAM10] "The State of the Internet, 3rd Quarter 2009", Akamai Technologies, Inc., January 2010.

[All00]   Allman, M., "A Web Server's View of the Transport Layer", ACM Computer Communication Review, 30(5), October 2000.

[Bel10]   Belshe, M., "A Client-Side Argument For Changing TCP Slow Start", January, 2010. URL http://sites.google.com/a/chromium.org/dev/spdy/An_Argument_For_Changing_TCP_Slow_Start.pdf

[Chu09]   Chu, J., "Tuning TCP Parameters for the 21st Century", Presented to 75th IETF TCPM working group meeting, July 2009. http://www.ietf.org/proceedings/75/slides/tcpm-1.pdf.

[DGHS07] Dischinger, M., Gummadi, K., Haeberlen, A. and S. Saroiu, "Characterizing Residential Broadband Networks", Internet

Measurement Conference, October 24-26, 2007.

[Duk10] Dukkipati, N., Refice, T., Cheng, Y., Chu, J., Sutin, N., Agarwal, A., Herbert, T. and J. Arvind, "An Argument for Increasing TCP's Initial Congestion Window", March, 2010. URL http://code.google.com/speed/articles/tcp_initcwnd_paper.pdf

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

[FJ93] Floyd, S. and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.

[IOR2009] Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J. Jahanian, F. and M. Karir, "Atlas Internet Observatory 2009 Annual Report", 47th NANOG Conference, October 2009.

[Jac88] Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.

[LAJW07] Liu, D., Allman, M., Jin, S. and L. Wang, "Congestion Control Without a Startup Phase", Protocols for Fast, Long Distance Networks (PFLDnet) Workshop, February 2007. URL http://www.icir.org/mallman/papers/jumpstart-pfldnet07.pdf

[PK98] Padmanabhan V.N. and R. Katz, "TCP Fast Start: A technique tbr speeding up web transfers", in Proceedings of IEEE Globecorn '98 Internet Mini-Conference, 1998.

[PRAKS02] Partridge, C., Rockwell, D., Allman, M., Krishnan, R. and J. Sterbenz, "A Swifter Start for TCP", Technical Report No. 8339, BBN Technologies, March 2002.

[PWSB09] Papadimitriou, D., Welzl, M., Scharf, M. and B. Briscoe, "Open Research Issues in Internet Congestion Control", section 3.4, Internet-draft draft-irtf-iccrg-welzl-congestion-control-open-research-05.txt, work in progress.

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

[RFC2414] Allman, M., Floyd, S. and C. Partridge, "Increasing TCP's Initial Window", RFC 2414, September 1998.

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

[RFC3042] Allman, M., Balakrishnan, H. and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January 2001.

[RFC3150] Dawkins, S., Montenegro, G., Kojo, M. and V. Magret, "End-to-end Performance Implications of Slow Links", RFC 3150, July 2001.

[RFC3782] Floyd, S., Henderson, T., and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 3782, April 2004.

[RFC4782] Floyd, S., Allman, M., Jain, A. and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, January 2007.

[RJ10] Ramachandran, S. and A. Jain, "Aggregate Statistics of Size Related Metrics of Web Pages metrics", 2010. URL http://code.google.com/speed/articles/web-metrics.html

[Sch08] Scharf, M., "Quick-Start, Jump-Start, and Other Fast Startup Approaches", November 17, 2008. URL http://www.ietf.org/old/2009/proceedings/08nov/slides/iccrg-2.pdf

[SPDY] "SPDY: An experimental protocol for a faster web", URL http://dev.chromium.org/spdy

[Ste08] Sounders S., "Roundup on Parallel Connections", High Performance Web Sites blog. URL http://www.stevesouders.com/blog/2008/03/20/roundup-on-parallel-connections

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

Author's Addresses

H.K. Jerry Chu
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA
EMail: hkchu@google.com

Nandita Dukkipati
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA
EMail: nanditad@google.com

Yuchung Cheng
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA
EMail: ycheng@google.com

Matt Mathis
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA
EMail: mattmathis@google.com

Acknowledgement

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