

Network Working Group
Request for Comments: 3148
Category: Informational

M. Mathis
Pittsburgh Supercomputing Center
M. Allman
BBN/NASA Glenn
July 2001

A Framework for Defining Empirical Bulk Transfer Capacity Metrics

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

Abstract

This document defines a framework for standardizing multiple BTC (Bulk Transport Capacity) metrics that parallel the permitted transport diversity.

1 Introduction

Bulk Transport Capacity (BTC) is a measure of a network's ability to transfer significant quantities of data with a single congestion-aware transport connection (e.g., TCP). The intuitive definition of BTC is the expected long term average data rate (bits per second) of a single ideal TCP implementation over the path in question. However, there are many congestion control algorithms (and hence transport implementations) permitted by IETF standards. This diversity in transport algorithms creates a difficulty for standardizing BTC metrics because the allowed diversity is sufficient to lead to situations where different implementations will yield non-comparable measures -- and potentially fail the formal tests for being a metric.

Two approaches are used. First, each BTC metric must be much more tightly specified than the typical IETF protocol. Second, each BTC methodology is expected to collect some ancillary metrics which are potentially useful to support analytical models of BTC.

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

[RFC2119] was written with protocols in mind, the key words are used in this document for similar reasons. They are used to ensure that each BTC methodology defined contains specific pieces of information.

Bulk Transport Capacity (BTC) is a measure of a network's ability to transfer significant quantities of data with a single congestion-aware transport connection (e.g., TCP). For many applications the BTC of the underlying network dominates the overall elapsed time for the application to run and thus dominates the performance as perceived by a user. Examples of such applications include FTP, and the world wide web when delivering large images or documents. The intuitive definition of BTC is the expected long term average data rate (bits per second) of a single ideal TCP implementation over the path in question. The specific definition of the bulk transfer capacity that MUST be reported by a BTC tool is:

$$\text{BTC} = \text{data_sent} / \text{elapsed_time}$$

where "data_sent" represents the unique "data" bits transferred (i.e., not including header bits or emulated header bits). Also note that the amount of data sent should only include the unique number of bits transmitted (i.e., if a particular packet is retransmitted the data it contains should be counted only once).

Central to the notion of bulk transport capacity is the idea that all transport protocols should have similar responses to congestion in the Internet. Indeed the only form of equity significantly deployed in the Internet today is that the vast majority of all traffic is carried by TCP implementations sharing common congestion control algorithms largely due to a shared developmental heritage.

[RFC2581] specifies the standard congestion control algorithms used by TCP implementations. Even though this document is a (proposed) standard, it permits considerable latitude in implementation. This latitude is by design, to encourage ongoing evolution in congestion control algorithms.

This legal diversity in congestion control algorithms creates a difficulty for standardizing BTC metrics because the allowed diversity is sufficient to lead to situations where different implementations will yield non-comparable measures -- and potentially fail the formal tests for being a metric.

There is also evidence that most TCP implementations exhibit non-linear performance over some portion of their operating region. It is possible to construct simple simulation examples where incremental improvements to a path (such as raising the link data rate) results in lower overall TCP throughput (or BTC) [Mat98].

We believe that such non-linearity reflects weakness in our current understanding of congestion control and is present to some extent in all TCP implementations and BTC metrics. Note that such non-linearity (in either TCP or a BTC metric) is potentially problematic in the market because investment in capacity might actually reduce the perceived quality of the network. Ongoing research in congestion dynamics has some hope of mitigating or modeling these non-linearities.

Related areas, including integrated services [RFC1633,RFC2216], differentiated services [RFC2475] and Internet traffic analysis [MSM097,PFTK98,Pax97b,LM97] are all currently receiving significant attention from the research community. It is likely that we will see new experimental congestion control algorithms in the near future. In addition, Explicit Congestion Notification (ECN) [RFC2481] is being tested for Internet deployment. We do not yet know how any of these developments might affect BTC metrics, and thus the BTC framework and metrics may need to be revisited in the future.

This document defines a framework for standardizing multiple BTC metrics that parallel the permitted transport diversity. Two approaches are used. First, each BTC metric must be much more tightly specified than the typical IETF transport protocol. Second, each BTC methodology is expected to collect some ancillary metrics which are potentially useful to support analytical models of BTC. If a BTC methodology does not collect these ancillary metrics, it should collect enough information such that these metrics can be derived (for instance a segment trace file).

As an example, the models in [PFTK98, MSM097, OKM96a, Lak94] all predict bulk transfer performance based on path properties such as loss rate and round trip time. A BTC methodology that also provides ancillary measures of these properties is stronger because agreement with the analytical models can be used to corroborate the direct BTC measurement results.

More importantly the ancillary metrics are expected to be useful for resolving disparity between different BTC methodologies. For example, a path that predominantly experiences clustered packet losses is likely to exhibit vastly different measures from BTC metrics that mimic Tahoe, Reno, NewReno, and SACK TCP algorithms [FF96]. The differences in the BTC metrics over such a path might be diagnosed by an ancillary measure of loss clustering.

There are some path properties which are best measured as ancillary metrics to a transport protocol. Examples of such properties include bottleneck queue limits or the tendency to reorder packets. These are difficult or impossible to measure at low rates and unsafe to measure at rates higher than the bulk transport capacity of the path.

It is expected that at some point in the future there will exist an A-frame [RFC2330] which will unify all simple path metrics (e.g., segment loss rates, round trip time) and BTC ancillary metrics (e.g., queue size and packet reordering) with different versions of BTC metrics (e.g., that parallel Reno or SACK TCP).

2 Congestion Control Algorithms

Nearly all TCP implementations in use today utilize the congestion control algorithms published in [Jac88] and further refined in [RFC2581]. In addition to using the basic notion of using an ACK clock, TCP (and therefore BTC) implements five standard congestion control algorithms: Congestion Avoidance, Retransmission timeouts, Slow-start, Fast Retransmit and Fast Recovery. All BTC implementations MUST implement slow start and congestion avoidance, as specified in [RFC2581] (with extra details also specified, as outlined below). All BTC methodologies SHOULD implement fast retransmit and fast recovery as outlined in [RFC2581]. Finally, all BTC methodologies MUST implement a retransmission timeout.

The algorithms specified in [RFC2581] give implementers some choices in the details of the implementation. The following is a list of details about the congestion control algorithms that are either underspecified in [RFC2581] or very important to define when constructing a BTC methodology. These details MUST be specifically defined in each BTC methodology.

- * [RFC2581] does not standardize a specific algorithm for increasing cwnd during congestion avoidance. Several candidate algorithms are given in [RFC2581]. The algorithm used in a particular BTC methodology MUST be defined.
- * [RFC2581] does not specify which cwnd increase algorithm (slow start or congestion avoidance) should be used when cwnd equals ssthresh. This MUST be specified for each BTC methodology.
- * [RFC2581] allows TCPs to use advanced loss recovery mechanism such as NewReno [RFC2582,FF96,Hoe96] and SACK-based algorithms [FF96,MM96a,MM96b]. If used in a BTC implementation, such an algorithm MUST be fully defined.

- * The actual segment size, or method of choosing a segment size (e.g., path MTU discovery [RFC1191]) and the number of header bytes assumed to be prepended to each segment MUST be specified. In addition, if the segment size is artificially limited to less than the path MTU this MUST be indicated.
- * TCP includes a retransmission timeout (RTO) to trigger retransmissions of segments that have not been acknowledged within an appropriate amount of time and have not been retransmitted via some more advanced loss recovery algorithm. A BTC implementation MUST include a retransmission timer. Calculating the RTO is subject to a number of details that MUST be defined for each BTC metric. In addition, a BTC metric MUST define when the clock is set and the granularity of the clock.

[RFC2988] specifies the behavior of the retransmission timer. However, there are several details left to the implementer which MUST be specified for each BTC metric defined.

Note that as new congestion control algorithms are placed on the standards track they may be incorporated into BTC metrics (e.g., the Limited Transmit algorithm [ABF00]). However, any implementation decisions provided by the relevant RFCs SHOULD be fully specified in the particular BTC metric.

3 Ancillary Metrics

The following ancillary metrics can provide additional information about the network and the behavior of the implemented congestion control algorithms in response to the behavior of the network path. It is RECOMMENDED that these metrics be built into each BTC methodology. Alternatively, it is RECOMMENDED that the BTC implementation provide enough information such that the ancillary metrics can be derived via post-processing (e.g., by providing a segment trace of the connection).

3.1 Congestion Avoidance Capacity

The "Congestion Avoidance Capacity" (CAC) metric is the data rate (bits per second) of a fully specified implementation of the Congestion Avoidance algorithm, subject to the restriction that the Retransmission Timeout and Slow-Start algorithms are not invoked. The CAC metric is defined to have no meaning across Retransmission Timeouts or Slow-Start periods (except the single segment Slow-Start that is permitted to follow recovery, as discussed in section 2).

In principle a CAC metric would be an ideal BTC metric, as it captures what should be TCP's steady state behavior. But, there is a rather substantial difficulty with using it as such. The Self-Clocking of the Congestion Avoidance algorithm can be very fragile, depending on the specific details of the Fast Retransmit, Fast Recovery or advanced recovery algorithms chosen. It has been found that timeouts and periods of slow start loss recovery are prevalent in traffic on the Internet [LK98,BPS+97] and therefore these should be captured by the BTC metric.

When TCP loses Self-Clock it is re-established through a retransmission timeout and Slow-Start. These algorithms nearly always require more time than Congestion Avoidance would have taken. It is easily observed that unless the network loses an entire window of data (which would clearly require a retransmit timeout) TCP likely missed some opportunity to safely transmit data. That is, if TCP experiences a timeout after losing a partial window of data, it must have received at least one ACK that was generated after some of the partial data was delivered, but did not trigger the transmission of new data. Recent research in congestion control (e.g., FACK [MM96a], NewReno [FF96,RFC2582], rate-halving [MSML99]) can be characterized as making TCP's Self-Clock more tenacious, while preserving fairness under adverse conditions. This work is motivated by how poorly current TCP implementations perform under some conditions, often due to repeated clock loss. Since this is an active research area, different TCP implementations have rather considerable differences in their ability to preserve Self-Clock.

3.2 Preservation of Self-Clock

Losing the ACK clock can have a large effect on the overall BTC, and the clock is itself fragile in ways that are dependent on the loss recovery algorithm. Therefore, the transition between timer driven and Self-Clocked operation SHOULD be instrumented.

3.2.1 Lost Transmission Opportunities

If the last event before a timeout was the receipt of an ACK that did not trigger a transmission, the possibility exists that an alternate congestion control algorithm would have successfully preserved the Self-Clock. A BTC SHOULD instrument key items in the BTC state (such as the congestion window) in the hopes that this may lead to further improvements in congestion control algorithms.

Note that in the absence of knowledge about the future, it is not possible to design an algorithm that never misses transmission opportunities. However, there are ever more subtle ways to gauge network state, and to estimate if a given ACK is likely to be the last.

3.2.2 Loosing an Entire Window

If an entire window of data (or ACKs) is lost, there will be no returning ACKs to clock out additional data. This condition can be detected if the last event before a timeout was a data transmission triggered by an ACK. The loss of an entire window of data/ACKs forces recovery to be via a Retransmission Timeout and Slow-Start.

Losing an entire window of data implies an outage with a duration at least as long as a round trip time. Such an outage can not be diagnosed with low rate metrics and is unsafe to diagnose at higher rates than the BTC. Therefore all BTC metrics SHOULD instrument and report losses of an entire window of data.

Note that there are some conditions, such as when operating with a very small window, in which there is a significant probability that an entire window can be lost through individual random losses (again highlighting the importance of instrumenting cwnd).

3.2.3 Heroic Clock Preservation

All algorithms that permit a given BTC to sustain Self-Clock when other algorithms might not, SHOULD be instrumented. Furthermore, the details of the algorithms used MUST be fully documented (as discussed in section 2).

BTC metrics that can sustain Self-Clock in the presence of multiple losses within one round trip SHOULD instrument the loss distribution, such that the performance of alternate congestion control algorithms may be estimated (e.g., Reno style).

3.2.4 False Timeouts

All false timeouts, (where the retransmission timer expires before the ACK for some previously transmitted data arrives) SHOULD be instrumented when possible. Note that depending upon how the BTC metric implements sequence numbers, this may be difficult to detect.

3.3 Ancillary Metrics Relating to Flow Based Path Properties

All BTC metrics provide unique vantage points for observing certain path properties relating to closely spaced packets. As in the case of RTT duration outages, these can be impossible to diagnose at low rates (less than 1 packet per RTT) and inappropriate to test at rates above the BTC of the network path.

All BTC metrics SHOULD instrument packet reordering. The frequency and distance out-of-sequence SHOULD be instrumented for all out-of-order packets. The severity of the reordering can be classified as one of three different cases, each of which SHOULD be reported.

Segments that are only slightly out-of-order should not trigger the fast retransmit algorithm, but they may affect the window calculation. BTC metrics SHOULD document how slightly out-of-order segments affect the congestion window calculation.

If segments are sufficiently out-of-order, the Fast Retransmit algorithm will be invoked in advance of the delayed packet's late arrival. These events SHOULD be instrumented. Even though the late arriving packet will complete recovery, the the window will still be reduced by half.

Under some rare conditions segments have been observed that are far out of order - sometimes many seconds late [Pax97b]. These SHOULD always be instrumented.

BTC implementations SHOULD instrument the maximum cwnd observed during congestion avoidance and slow start. A TCP running over the same path as the BTC metric must have sufficient sender buffer space and receiver window (and window shift [RFC1323]) to cover this cwnd in order to expect the same performance.

There are several other path properties that one might measure within a BTC metric. For example, with an embedded one-way delay metric it may be possible to measure how queuing delay and and (RED) drop probabilities are correlated to window size. These are open research questions.

3.4 Ancillary Metrics as Calibration Checks

Unlike low rate metrics, BTC SHOULD include explicit checks that the test platform is not the bottleneck.

Any detected dropped packets within the sending host MUST be reported. Unless the sending interface is the path bottleneck, any dropped packets probably indicates a measurement failure.

The maximum queue lengths within the sending host SHOULD be instrumented. Any significant queue may indicate that the sending host has insufficient burst data rate, and is smoothing the data being transmitted into the network.

3.5 Ancillary Metrics Relating to the Need for Advanced TCP Features

If TCP would require advanced TCP extensions to match BTC performance (such as RFC 1323 or RFC 2018 features), it SHOULD be reported.

3.6 Validate Reverse Path Load

To the extent possible, the BTC metric SHOULD distinguish between the properties of the forward and reverse paths.

BTC methodologies which rely on non-cooperating receivers may only be able to measure round trip path properties and may not be able to independently differentiate between the properties of the forward and reverse paths. In this case the load on the reverse path contributed by the BTC metric SHOULD be instrumented (or computed) to permit other means of gauge the proportion of the round trip path properties attributed to the the forward and reverse paths.

To the extent possible, BTC methodologies that rely on cooperating receivers SHOULD support separate ancillary metrics for the forward and reverse paths.

4 Security Considerations

Conducting Internet measurements raises security concerns. This memo does not specify a particular implementation of a metric, so it does not directly affect the security of the Internet nor of applications which run on the Internet. However, metrics produced within this framework, and in particular implementations of the metrics may create security issues.

4.1 Denial of Service Attacks

Bulk Transport Capacity metrics, as defined in this document, naturally attempt to fill a bottleneck link. The BTC metrics based on this specification will be as "network friendly" as current well-tuned TCP connections. However, since the "connection" may not be using TCP packets, a BTC test may appear to network operators as a denial of service attack.

Administrators of the source host of a test, the destination host of a test, and the intervening network(s) may wish to establish bilateral or multi-lateral agreements regarding the timing, size, and frequency of collection of BTC metrics.

4.2 User data confidentiality

Metrics within this framework generate packets for a sample, rather than taking samples based on user data. Thus, a BTC metric does not threaten user data confidentiality.

4.3 Interference with metrics

It may be possible to identify that a certain packet or stream of packets are part of a BTC metric. With that knowledge at the destination and/or the intervening networks, it is possible to change the processing of the packets (e.g., increasing or decreasing delay, introducing or heroically preventing loss) that may distort the measured performance. It may also be possible to generate additional packets that appear to be part of a BTC metric. These additional packets are likely to perturb the results of the sample measurement.

To discourage the kind of interference mentioned above, packet interference checks, such as cryptographic hash, may be used.

5 IANA Considerations

Since this metric framework does not define a specific protocol, nor does it define any well-known values, there are no IANA considerations for this document. However, a bulk transport capacity metric within this framework, and in particular protocols that implement a metric may have IANA considerations that need to be addressed.

6 Acknowledgments

Thanks to Wil Leland, Jeff Semke, Matt Zekauskas and the IPPM working group for numerous clarifications.

Matt Mathis's work was supported by the National Science Foundation under Grant Numbers 9415552 and 9870758.

7 References

- [BPS+97] Hari Balakrishnan, Venkata Padmanabhan, Srinivasan Seshan, Mark Stemm, and Randy Katz. TCP Behavior of a Busy Web Server: Analysis and Improvements. Technical Report UCB/CSD-97-966, August 1997. Available from <http://nms.lcs.mit.edu/~hari/papers/csd-97-966.ps>. (Also in Proc. IEEE INFOCOM Conf., San Francisco, CA, March 1998.)
- [FF96] Fall, K., Floyd, S.. "Simulation-based Comparisons of Tahoe, Reno and SACK TCP". Computer Communication Review, July 1996. <ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z>.
- [Flo95] Floyd, S., "TCP and successive fast retransmits", March 1995, Obtain via <ftp://ftp.ee.lbl.gov/papers/fastretrans.ps>.
- [Hoe96] Hoe, J., "Improving the start-up behavior of a congestion control scheme for TCP, Proceedings of ACM SIGCOMM '96, August 1996.
- [Hoe95] Hoe, J., "Startup dynamics of TCP's congestion control and avoidance schemes". Master's thesis, Massachusetts Institute of Technology, June 1995.
- [Jac88] Jacobson, V., "Congestion Avoidance and Control", Proceedings of SIGCOMM '88, Stanford, CA., August 1988.
- [Lak94] V. T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. IFIP Transactions C-26, High Performance Networking, pages 135--150, 1994.
- [LK98] Lin, D. and Kung, H.T., "TCP Fast Recovery Strategies: Analysis and Improvements", Proceedings of InfoCom, March 1998.
- [LM97] T.V.Lakshman and U.Madhow. "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss". IEEE/ACM Transactions on Networking, Vol. 5, No. 3, June 1997, pp.336-350.

- [Mat98] Mathis, M., "Empirical Bulk Transfer Capacity", IP Performance Metrics Working Group report in Proceedings of the Forty Third Internet Engineering Task Force, Orlando, FL, December 1988. Available from <http://www.ietf.org/proceedings/98dec/43rd-ietf-98dec-122.html> and <http://www.ietf.org/proceedings/98dec/slides/ippm-mathis-98dec.pdf>.
- [MM96a] Mathis, M. and Mahdavi, J. "Forward acknowledgment: Refining TCP congestion control", Proceedings of ACM SIGCOMM '96, Stanford, CA., August 1996.
- [MM96b] M. Mathis, J. Mahdavi, "TCP Rate-Halving with Bounding Parameters". Available from <http://www.psc.edu/networking/papers/FACKnotes/current>.
- [MSML99] Mathis, M., Semke, J., Mahdavi, J., Lahey, K., "The Rate-Halving Algorithm for TCP Congestion Control", June 1999, Work in Progress.
- [MSMO97] Mathis, M., Semke, J., Mahdavi, J., Ott, T., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communications Review, 27(3), July 1997.
- [OKM96a], Ott, T., Kemperman, J., Mathis, M., "The Stationary Behavior of Ideal TCP Congestion Avoidance", In progress, August 1996. Obtain via pub/tjo/TCPwindow.ps using anonymous ftp to <ftp.bellcore.com>
- [OKM96b], Ott, T., Kemperman, J., Mathis, M., "Window Size Behavior in TCP/IP with Constant Loss Probability", DIMACS Special Year on Networks, Workshop on Performance of Real-Time Applications on the Internet, Nov 1996.
- [Pax97a] Paxson, V., "Automated Packet Trace Analysis of TCP Implementations", Proceedings of ACM SIGCOMM '97, August 1997.
- [Pax97b] Paxson, V., "End-to-End Internet Packet Dynamics," Proceedings of SIGCOMM '97, Cannes, France, Sep. 1997.
- [PFTK98] Padhye, J., Firoiu, V., Towsley, D., and Kurose, J., "TCP Throughput: A Simple Model and its Empirical Validation", Proceedings of ACM SIGCOMM '98, August 1998.

- [RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981. Obtain via: <http://www.rfc-editor.org/rfc/rfc793.txt>
- [RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990. Obtain via: <http://www.rfc-editor.org/rfc/rfc1191.txt>
- [RFC1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", May 1992. Obtain via: <http://www.rfc-editor.org/rfc/rfc1323.txt>
- [RFC1633] Braden R., Clark D. and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994. Obtain via: <http://www.rfc-editor.org/rfc/rfc1633.txt>
- [RFC2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, January 1997. Obtain via: <http://www.rfc-editor.org/rfc/rfc2001.txt>
- [RFC2018] Mathis, M., Mahdavi, J. Floyd, S., Romanow, A., "TCP Selective Acknowledgment Options", RFC 2018, October 1996. Obtain via: <http://www.rfc-editor.org/rfc/rfc2018.txt>
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. Obtain via: <http://www.rfc-editor.org/rfc/rfc2119.txt>
- [RFC2216] Shenker, S. and J. Wroclawski, "Network Element Service Specification Template", RFC 2216, September 1997. Obtain via: <http://www.rfc-editor.org/rfc/rfc2216.txt>
- [RFC2330] Paxson, V., Almes, G., Mahdavi, J. and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, April 1998. Obtain via: <http://www.rfc-editor.org/rfc/rfc2330.txt>
- [RFC2475] Black D., Blake S., Carlson M., Davies E., Wang Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998. Obtain via: <http://www.rfc-editor.org/rfc/rfc2475.txt>

- [RFC2481] Ramakrishnan, K. and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, January 1999. Obtain via: <http://www.rfc-editor.org/rfc/rfc2481.txt>
- [RFC2525] Paxson, V., Allman, M., Dawson, S., Fenner, W., Griner, J., Heavens, I., Lahey, K., Semke, J. and B. Volz, "Known TCP Implementation Problems", RFC 2525, March 1999. Obtain via: <http://www.rfc-editor.org/rfc/rfc2525.txt>
- [RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999. Obtain via: <http://www.rfc-editor.org/rfc/rfc2581.txt>
- [RFC2582] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999. Obtain via: <http://www.rfc-editor.org/rfc/rfc2582.txt>
- [RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000. Obtain via: <http://www.rfc-editor.org/rfc/rfc2988.txt>
- [RFC3042] Allman, M., Balakrishnan, H. and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January 2001. Obtain via: <http://www.rfc-editor.org/rfc/rfc3042.txt>
- [Ste94] Stevens, W., "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.
- [WS95] Wright, G., Stevens, W., "TCP/IP Illustrated Volume II: The Implementation", Addison-Wesley, 1995.

Author's Addresses

Matt Mathis
Pittsburgh Supercomputing Center
4400 Fifth Ave.
Pittsburgh PA 15213

EMail: mathis@psc.edu
<http://www.psc.edu/~mathis>

Mark Allman
BBN Technologies/NASA Glenn Research Center
Lewis Field
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135

Phone: 216-433-6586
EMail: mallman@bbn.com
<http://roland.grc.nasa.gov/~mallman>

Full Copyright Statement

Copyright (C) The Internet Society (2001). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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

