Network Working Group Matt Mathis
INTERNET-DRAFT Pittsburgh Supercomputing Center
Expiration Date: Jan. 2000 Mark Allman
NASA Glenn
June, 1999

Empirical Bulk Transfer Capacity

< draft-ietf-ippm-btc-framework-01.txt >

Status of this Document

Abstract

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.

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 protocol. Pseudo-code or reference implementations are expected to be the norm. Second, each BTC methodology is expected to collect some ancillary metrics which are potentially useful to support analytical models of BTC.

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). 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.

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 these 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 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.

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 [MathisIPPM1998?].

We beleive 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 preceived quality of the network. Ongoing research in congestion dynamics has some hope of mitigating or modeling the these non-linearities.

Furthermore related areas, including Integrated services[@@], differentiated services[@@] and Internet traffic analysis[@@] 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.

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. Pseudo-code or reference implementations are expected to be the norm. 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).

For example, the models in [PFTK98, MSMO97, 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 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 use these algorithms as they are defined in [RFC2581] (which the reader is assumed to be familiar with). However, in all cases a BTC metric must more tightly specify these algorithms, as discussed below.

2.1 Congestion Avoidance

The Congestion Avoidance algorithm drives the steady-state bulk transfer behavior of TCP. It calls for opening the congestion window (cwnd) by a constant additive amount during each round trip time (RTT), and closing cwnd by a constant multiplicative fraction on congestion, as indicated by lost segments or Explicit Congestion Notification messages [RFC2481]. The window closes by half the number of outstanding data segments in flight when loss is detected. A BTC metric must specify the following Congestion Avoidance details:

The exact algorithm for incrementing cwnd in TCP is left to the implementer. Several candidate algorithms are outlined in [RFC2581]. In addition, some of these algorithms include some rounding. For these reasons, the exact algorithm for increasing cwnd during congestion avoidance must be fully specified for each BTC metric defined.

[RFC2581] permits, but does not require, an extra plus one segment cwnd adjustment following the multiplicative decrease of cwnd. This is because [RFC2581] allows a single invocation of the Slow-Start algorithm when when cwnd equals ssthresh at the end of recovery.

2.2 Retransmission Timeouts

In order to provide reliable data delivery, TCP resends a segment if the ACK for the given segment does not arrive before the retransmission timer (RTO) expires. A BTC metric must implement an RTO timer to trigger retransmissions not handled by the fast retransmit algorithm. Such retransmissions can have a large impact on the measured BTC of the path. Calculating the RTO is subject to a number of details that are not standardized (however, [WS95] outlines a popular implementation). When implementing a BTC metric the details of the RTO calculation, how and when the clock is set, as well as the clock granularity must be fully documented.

2.3 Slow Start

Slow start is part of TCP's transient behavior. It is used to quickly increase the congestion window for new or recently restarted connections up to an appropriate level for the network path. In addition, slow start is used to restart the ACK clock after a retransmission timeout. A BTC implementation must use the slow start algorithm, as specified by [RFC2581]. The slow start algorithm is used while the congestion window (cwnd) is less than the slow start threshold (ssthresh). However, whether to use slow start or congestion avoidance when cwnd equals ssthresh is left to the implementer by [RFC2581]. This detail must be specified in every specific BTC metric definition.

2.4 Fast Retransmit/Fast Recovery

The Fast Retransmit/Fast Recovery algorithms are used to infer segment loss before the RTO expires. A BTC implementation must implement the algorithms as defined in [RFC2581].

In Reno TCP, Fast Retransmit and Fast Recovery are used to support the Congestion Avoidance algorithm during loss recovery. During Fast Recovery, the data receiver sends duplicated acknowledgments, per the TCP specification [RFC793]. The data sender uses these duplicate ACKs to detect loss, to estimate the quantity of outstanding data in the network and to clock out new data in an effort to keep the ACK clock running.

The Fast Retransmit/Fast Recovery algorithms should be implemented in all BTC methodologies as specified in [RFC2581].

2.5 Advanced Recovery Algorithms

It has been observed that under some conditions the Fast Retransmit and Fast Recovery algorithms do not reliably preserve TCP's Self-Clock, causing unpredictable or unstable TCP performance [Lak94@@@check, Flo95]. Simulations of reference TCP implementations have uncovered situations where incidental changes in the network path have a large effect on performance [MM96a]. Additional simulations have shown that under some conditions, slightly better networks (higher bandwidth, lower delay or less competing traffic) yield lower throughput [MathisIPPMDec1998?].

[RFC2581] allows a TCP implementation to use more robust loss recovery algorithms, such as NewReno [RFC2582,FF96,Hoe96] and SACK-based algorithms [FF96,MM96a,MM96b]. While allowing these algorithms, [RFC2581] does not define any such algorithm and therefore, a BTC metric that implements advanced loss recovery algorithms must fully specify the details.

2.6 Segment Size

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 (if known).

3 Ancillary Metrics

The following ancillary metrics can provide additional information about the network and the behavior of the implemented congestion control algorithm in response to the behavior of the network path. It is recommended that these metrics be built into each BTC methodology. Alternatively, the BTC implementation should 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 2.3).

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 above. It has been found that timeouts and periods of slow start loss recovery are prevalent in traffic on the Internet [LK98] and therefore these should be included in the BTC metric.

When TCP looses Self-Clock it is reestablished 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 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]) can be characterized as making TCP's Self-Clock more tenacious, while preserving fairness under adverse conditions. This work is often 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, it is important that the transition between timer driven and Self-Clocked operation 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 retransmission, the possibility exists that an alternate congestion control algorithm would have successfully preserved the Self-Clock. In this event, instrumenting key parts of the BTC state (such as the congestion window) 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 at 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.

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.

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 Reno style bulk transport can be estimated.

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 instrumenting 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.

All BTC metrics should instrument packet reordering. The frequency and distance out of sequence must 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.

> Packets that are only slightly out of order should not trigger retransmission (via fast retransmit), but they may affect the window calculation. BTC metrics must document how slightly out-of-order packets affect the congestion window calculation.
>
> If packets are sufficiently out-of-order, the Fast Retransmit algorithm will be invoked in advance of the delayed packet's late arrival. These events must be instrumented. Even though the the late arriving packet will complete recovery, the the window will still be reduced by half.
>
> Under some rare conditions packets have been observed that are far out of order - sometimes many seconds late [Pax97b]. These should always be instrumented.

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

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 queueing delay and and (RED) drop probabilities are correlated to window size. These are open research questions.

3.4 Ancillary Metrics Pertaining to MTU Discovery

Under some conditions, BTC can be very sensitive to segment size. In addition to instrumenting the segment size, a BTC metric should indicate how it was selected: by path MTU discovery [RFC1191], a manual configuration, system default, or the maximum MTU for the interface.

Note that the most popular LAN technologies have smaller MTUs than nearly all WAN technologies. As a consequence, it is difficult to measure the true performance of a wide area path without subjecting it to the smaller MTU of the LAN.

3.4 Ancillary Metrics as Calibration Checks

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

Ideally all queues within the tester should be instrumented. All packets dropped within the tester should be instrumented as tester failures, invalidating a measurement.

The maximum queue lengths should be instrumented. Any significant queue may indicate that the tester itself has insufficient burst data rate, and is slightly smoothing the data into the network.

3.4.3 Validate Reverse path load

@@@@ What happens to a BTC when the reverse path is congested? Is this identical to TCP? What should happen? How should it be instrumented?

```
#
# Some implementations (mine!) have an annoying feature whereby ACK loss
# looks just like data loss.  This should be documented.  If ACK loss
# and data loss can be detected separately, I think ACK loss rate should
# be reported, as it slightly changes the ACK clock (can impact
# algorithms like slow start that work on a per ACK basis and can make
# the sender more bursty, which could cause more loss).
@ and mine --MM--
```

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.

4 Acknowledgments

Thanks to Jeff Semke for numerous clarifications.

5 References

[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] Lakshman, Effects of random loss

[LK98] Lin, D. and Kung, H.T., "TCP Fast Recovery Strategies: Analysis and Improvements", Proceedings of InfoCom, March 1998.

[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.

[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", 1981, Obtain via: ftp://ds.internic.net/rfc/rfc793.txt

[RFC1191] Mogul, J., Deering, S., "Path MTU Discovery", November 1990, Obtain via: ftp://ds.internic.net/rfc/rfc1191.txt

[RFC1323] Jacobson, V., Braden, R., Borman, D., "TCP Extensions for High Performance", May 1992, Obtain via: ftp://ds.internic.net/rfc/rfc1323.txt

[RFC2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", 1997, Obtain via: ftp://ds.internic.net/rfc/rfc2001.txt

[RFC2018] Mathis, M., Mahdavi, J. Floyd, S., Romanow, A., "TCP Selective Acknowledgment Options", 1996, Obtain via: ftp://ds.internic.net/rfc/rfc2018.txt

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., Mathis, M., "Framework for IP Performance Metrics" , 1998, Obtain via: ftp://ds.internic.net/rfc/rfc2330.txt

[RFC2481] K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", 1999, Obtain via: ftp://ds.internic.net/rfc/rfc2481.txt

[RFC2525] V. Paxson, M. Allman, S. Dawson, W. Fenner, J. Griner, I. Heavens, K. Lahey, J. Semke, B. Volz, "Known TCP Implementation Problems", 1999, Obtain via: ftp://ds.internic.net/rfc/rfc2525.txt

[RFC2581] Allman, M., Paxson, V., Stevens, W., "TCP Congestion Control"., 1999, Obtain via: ftp://ds.internic.net/rfc/rfc2581.txt

[RFC2582] Floyd, S., Henderson, T., "The NewReno Modification to TCP's Fast Recovery Algorithm", 1999, Obtain via: ftp://ds.internic.net/rfc/rfc2582.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
mathis@psc.edu
http://www.psc.edu/~mathis

Mark Allman
NASA Glenn Research Center/GTE Internetworking
Lewis Field
21000 Brookpark Rd. MS 54-2
Cleveland, OH 44135
216-433-6586
mallman@grc.nasa.gov
http://roland.grc.nasa.gov/~mallman