IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

M. Mathis
Google, Inc
A. Morton
AT&T Labs
Oct 19, 2015

Model Based Metrics for Bulk Transport Capacity
draft-ietf-ippm-model-based-metrics-07.txt

Abstract

We introduce a new class of Model Based Metrics designed to assess if a complete Internet path can be expected to meet a predefined Target Transport Performance by applying a suite of IP diagnostic tests to successive subpaths. The subpath-at-a-time tests can be robustly applied to key infrastructure, such as interconnects or even individual devices, to accurately detect if any part of the infrastructure will prevent paths traversing it from meeting the Target Transport Performance.

For Bulk Transport Capacity, the IP diagnostics are built on test streams that mimic TCP over the complete path and statistical criteria for evaluating the packet transfer statistics of those streams. The temporal structure of the test stream (bursts, etc) mimic TCP or other transport protocol carrying bulk data over a long path but are constructed to be independent of the details of the subpath under test, end systems or applications. Likewise the success criteria evaluates the packet transfer statistics of the subpath against criteria determined by protocol performance models applied to the Target Transport Performance of the complete path. The success criteria also does not depend on the details of the subpath, end systems or application.

Model Based Metrics exhibit several important new properties not present in other Bulk Transport Capacity Metrics, including the ability to reason about concatenated or overlapping subpaths. The results are vantage independent which is critical for supporting independent validation of tests by comparing results from multiple measurement points.

This document does not define the IP diagnostic tests, but provides a framework for designing suites of IP diagnostic tests that are tailored to confirming that infrastructure can meet the predetermined Target Transport Performance.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Model Based Metrics (MBM) rely on mathematical models to specify a Targeted Suite of IP Diagnostic tests, designed to assess whether common transport protocols can be expected to meet a predetermined Target Transport Performance over an Internet path. This note describes the modeling framework to derive the test parameters for accessing an Internet path's ability to support a predetermined Bulk Transport Capacity.

Each test in the Targeted IP Diagnostic Suite (TIDS) measures some aspect of IP packet transfer needed to meet the Target Transport Performance. For Bulk Transport Capacity the TIDS includes IP diagnostic tests to verify that there is: sufficient IP capacity (data rate); sufficient queue space at bottlenecks to absorb and deliver typical transport bursts; and that the background packet loss ratio is low enough not to interfere with congestion control; and other properties described below. Unlike typical IPPM metrics which yield measures of network properties, Model Based Metrics nominally yield pass/fail evaluations of the ability of standard transport protocols to meet the specific performance objective over some network path.

In most cases the IP diagnostic tests can be implemented by combining existing IPPM metrics with additional controls for generating test streams having a specified temporal structure (busts or standing queues, etc) and statistical criteria for evaluating packet transfer. The temporal structure of the test streams mimic transport protocol behavior over the complete path, the statistical criteria models the transport protocol's response to less than ideal IP packet transfer.

This note describes an alternative to the approach presented in "A Framework for Defining Empirical Bulk Transfer Capacity Metrics" [RFC3148]. In the future, other Model Based Metrics may cover other applications and transports, such as VoIP over RTP.

The MBM approach, mapping Target Transport Performance to a Targeted IP Diagnostic Suite (TIDS) of IP tests, solves some intrinsic problems with using TCP or other throughput maximizing protocols for measurement. In particular all throughput maximizing protocols (and TCP congestion control in particular) cause some level of congestion in order to detect when they have filled the network. This self inflicted congestion obscures the network properties of interest and introduces non-linear equilibrium behaviors that make any resulting measurements useless as metrics because they have no predictive value for conditions or paths different than that of the measurement itself. These problems are discussed at length in Section 4.

A Targeted IP Diagnostic Suite does not have such difficulties. IP diagnostics can be constructed such that they make strong statistical statements about path properties that are independent of the measurement details, such as vantage and choice of measurement points. Model Based Metrics are designed to bridge the gap between empirical IP measurements and expected TCP performance.

1.1. Version Control

RFC Editor: Please remove this entire subsection prior to publication.

Please send comments about this draft to ippm@ietf.org. See http://goo.gl/02tkD for more information including: interim drafts, an up to date todo list and information on contributing.

Formatted: Mon Oct 19 15:59:51 PDT 2015

Changes since -06 draft:

- More language nits:
  - "Targeted IP Diagnostic Suite (TIDS)" replaces "Targeted Diagnostic Suite (TDS)".
  - "implied bottleneck IP capacity" replaces "implied bottleneck IP rate".
  - Updated to ECN CE Marks.
  - Added "specified temporal structure"
  - "test stream" replaces "test traffic"
  - "packet transfer" replaces "packet delivery"
  - Reworked discussion of slowstart, bursts and pacing.
  - RFC 7567 replaces RFC 2309.

Changes since -05 draft:

- Wordsmithing on sections overhauled in -05 draft.
- Reorganized the document:
  - Relocated subsection "Preconditions".
  - Relocated subsection "New Requirements relative to RFC 2330".
- Addressed nits and not so nits by Ruediger Geib. (Thanks!)
- Substantially tightened the entire definitions section.
- Many terminology changes, to better conform to other docs :
  - IP rate and IP capacity (following RFC 5136) replaces various forms of link data rate.
  - subpath replaces link.
  - target_window_size replaces target_pipe_size.
  - implied bottleneck IP rate replaces effective bottleneck link rate.
  - Packet delivery statistics replaces delivery statistics.

Changes since -04 draft:

- The introduction was heavily overhauled: split into a separate introduction and overview.
- The new shorter introduction:
  - Is a problem statement;
  - This document provides a framework;
  - That it replaces TCP measurement by IP tests;
  - That the results are pass/fail.
- Added a diagram of the framework to the overview
- and introduces all of the elements of the framework.
- Renumbered sections, reducing the depth of some section numbers.
- Updated definitions to better agree with other documents:
  - Reordered section 2
  - Bulk [data] performance -> Bulk Transport Capacity, everywhere including the title.
  - loss rate and loss probability -> packet loss ratio
  - end-to-end path -> complete path
  - [end-to-end][target] performance -> Target Transport Performance
  - load test -> capacity test

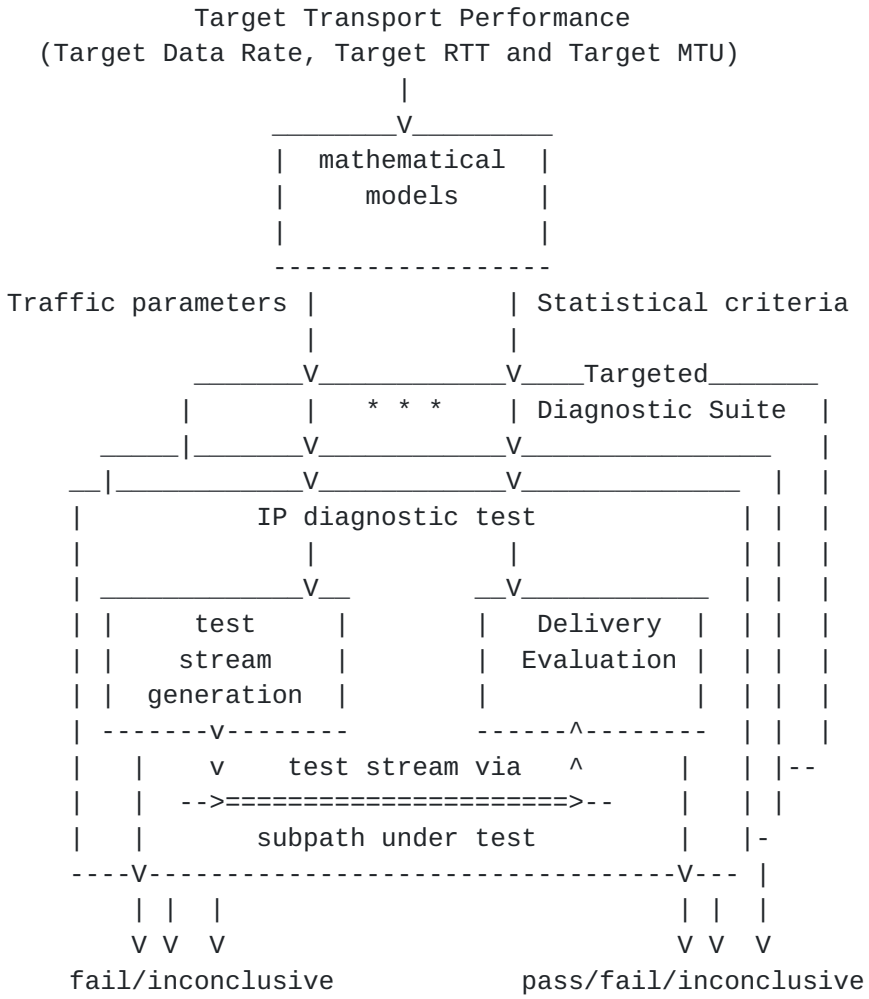
2. Overview

This document describes a modeling framework for deriving a Targeted IP Diagnostic Suite from a predetermined Target Transport Performance. It is not a complete specification, and relies on other standards documents to define important details such as packet type-p selection, sampling techniques, vantage selection, etc. We imagine Fully Specified Targeted IP Diagnostic Suites (FSTIDS), that define all of these details. We use Targeted IP Diagnostic Suite (TIDS) to refer to the subset of such a specification that is in scope for this document. This terminology is defined in Section 3.

Section 4 describes some key aspects of TCP behavior and what they imply about the requirements for IP packet transfer. Most of the IP diagnostic tests needed to confirm that the path meets these properties can be built on existing IPPM metrics, with the addition of statistical criteria for evaluating packet transfer and in a few cases, new mechanisms to implement the required temporal structure. (One group of tests, the standing queue tests described in Section 8.2, don't correspond to existing IPPM metrics, but suitable metrics can be patterned after existing tools.)

Figure 1 shows the MBM modeling and measurement framework. The Target Transport Performance, at the top of the figure, is determined by the needs of the user or application, outside the scope of this document. For Bulk Transport Capacity, the main performance parameter of interest is the Target Data Rate. However, since TCP's

ability to compensate for less than ideal network conditions is fundamentally affected by the Round Trip Time (RTT) and the Maximum Transmission Unit (MTU) of the complete path, these parameters must also be specified in advance based on knowledge about the intended application setting. They may reflect a specific application over real path through the Internet or an idealized application and hypothetical path representing a typical user community. Section 5 describes the common parameters and models derived from the Target Transport Performance.



Overall Modeling Framework

Figure 1

The mathematical models are used to design traffic patterns that mimic TCP or other transport protocol delivering bulk data and operating at the Target Data Rate, MTU and RTT over a full range of conditions, including flows that are bursty at multiple time scales. The traffic patterns are generated based on the three target

parameters of complete path and independent of the properties of individual subpaths using the techniques described in Section 6. As much as possible the measurement traffic is generated deterministically (precomputed) to minimize the extent to which test methodology, measurement points, measurement vantage or path partitioning affect the details of the measurement traffic.

Section 7 describes packet transfer statistics and methods test them against the bounds provided by the mathematical models. Since these statistics are typically the composition of subpaths of the complete path [RFC6049] , in situ testing requires that the end-to-end statistical bounds be apportioned as separate bounds for each subpath. Subpaths that are expected to be bottlenecks may be expected to contribute a larger fraction of the total packet loss. In compensation, non-bottlenecked subpaths have to be constrained to contribute less packet loss. The criteria for passing each test of a TIDS is an apportioned share of the total bound determined by the mathematical model from the Target Transport Performance.

Section 8 describes the suite of individual tests needed to verify all of required IP delivery properties. A subpath passes if and only if all of the individual IP diagnostic tests pass. Any subpath that fails any test indicates that some users are likely fail to attain their Target Transport Performance under some conditions. In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons including: the precomputed traffic pattern was not accurately generated; the measurement results were not statistically significant; and others such as failing to meet some required test preconditions. If all tests pass, except some are inconclusive then the entire suite is deemed to be inconclusive.

In Section 9 we present an example TIDS that might be representative of HD video, and illustrate how Model Based Metrics can be used to address difficult measurement situations, such as confirming that intercarrier exchanges have sufficient performance and capacity to deliver HD video between ISPs.

Since there is some uncertainty in the modeling process, Section 10 describes a validation procedure to diagnose and minimize false positive and false negative results.

3. 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 [RFC2119].

Note that terms containing underscores (rather than spaces) appear in equations in the modeling sections. In some cases both forms are used for aesthetic reasons, they do not have different meanings.

General Terminology:

Target: A general term for any parameter specified by or derived from the user's application or transport performance requirements.

Target Transport Performance: Application or transport performance goals for the complete path. For Bulk Transport Capacity defined in this note the Target Transport Performance includes the Target Data Rate, Target RTT and Target MTU as described below.

Target Data Rate: The specified application data rate required for an application's proper operation. Conventional BTC metrics are focused on the Target Data Rate, however these metrics had little or no predictive value because they do not consider the effects of the other two parameters of the Target Transport Performance, the RTT and MTU of the complete paths.

Target RTT (Round Trip Time): The specified baseline (minimum) RTT of the longest complete path over which the user expects to be able meet the target performance. TCP and other transport protocol's ability to compensate for path problems is generally proportional to the number of round trips per second. The Target RTT determines both key parameters of the traffic patterns (e.g. burst sizes) and the thresholds on acceptable IP packet transfer statistics. The Target RTT must be specified considering appropriate packets sizes: MTU sized packets on the forward path, ACK sized packets (typically header_overhead) on the return path. Note that Target RTT is specified and not measured, it determines the applicability of MBM measuremets for paths that are different than the measured path.

Target MTU (Maximum Transmission Unit): The specified maximum MTU supported by the complete path the over which the application expects to meet the target performance. Assume 1500 Byte MTU unless otherwise specified. If some subpath has a smaller MTU, then it becomes the Target MTU for the complete path, and all model calculations and subpath tests must use the same smaller MTU.

Targeted IP Diagnostic Suite (TIDS): A set of IP diagnostic tests designed to determine if an otherwise ideal complete path containing the subpath under test can sustain flows at a specific target_data_rate using target_MTU sized packets when the RTT of the complete path is target_RTT.

Fully Specified Targeted IP Diagnostic Suite: A TIDS together with additional specification such as "type-p", etc which are out of scope for this document, but need to be drawn from other standards documents.

Bulk Transport Capacity: Bulk Transport Capacity Metrics evaluate an Internet path's ability to carry bulk data, such as large files, streaming (non-real time) video, and under some conditions, web images and other content. Prior efforts to define BTC metrics have been based on [RFC3148], which predates our understanding of TCP and the requirements described in Section 4

IP diagnostic tests: Measurements or diagnostics to determine if packet transfer statistics meet some precomputed target.

traffic patterns: The temporal patterns or burstiness of traffic generated by applications over transport protocols such as TCP. There are several mechanisms that cause bursts at various time scales as described in Section 4.1. Our goal here is to mimic the range of common patterns (burst sizes and rates, etc), without tying our applicability to specific applications, implementations or technologies, which are sure to become stale.

packet transfer statistics: Raw, detailed or summary statistics about packet transfer properties of the IP layer including packet losses, ECN Congestion Experienced (CE) marks, reordering, or any other properties that may be germane to transport performance.

packet loss ratio: As defined in [I-D.ietf-ippm-2680-bis].

apportioned: To divide and allocate, for example budgeting packet loss across multiple subpaths such that the losses will accumulate to less than a specified end-to-end loss ratio. Apportioning metrics is essentially the inverse of the process described in [RFC5835].

open loop: A control theory term used to describe a class of techniques where systems that naturally exhibit circular dependencies can be analyzed by suppressing some of the dependencies, such that the resulting dependency graph is acyclic.

Terminology about paths, etc. See [RFC2330] and [RFC7398].

data sender: Host sending data and receiving ACKs.

data receiver: Host receiving data and sending ACKs.

complete path: The end-to-end path from the data sender to the data receiver.

subpath: A portion of the complete path. Note that there is no requirement that subpaths be non-overlapping. A subpath can be a small as a single device, link or interface.

measurement point: Measurement points as described in [RFC7398].

test path: A path between two measurement points that includes a subpath of the complete path under test. If the measurement points are off path, the test path may include "test leads" between the measurement points and the subpath.

dominant bottleneck: The bottleneck that generally determines most of packet transfer statistics for the entire path. It typically determines a flow's self clock timing, packet loss and ECN Congestion Experienced (CE) marking rate, with other potential bottlenecks having less effect on the packet transfer statistics. See Section 4.1 on TCP properties.

front path: The subpath from the data sender to the dominant bottleneck.

back path: The subpath from the dominant bottleneck to the receiver.

return path: The path taken by the ACKs from the data receiver to the data sender.

cross traffic: Other, potentially interfering, traffic competing for network resources (bandwidth and/or queue capacity).

Properties determined by the complete path and application. These are described in more detail in Section 5.1.

Application Data Rate: General term for the data rate as seen by the application above the transport layer in bytes per second. This is the payload data rate, and explicitly excludes transport and lower level headers (TCP/IP or other protocols), retransmissions and other overhead that is not part to the total quantity of data delivered to the application.

IP rate: The actual number of IP-layer bytes delivered through a subpath, per unit time, including TCP and IP headers, retransmits and other TCP/IP overhead. Follows from IP-type-P Link Usage [RFC5136].

IP capacity: The maximum number of IP-layer bytes that can be transmitted through a subpath, per unit time, including TCP and IP headers, retransmits and other TCP/IP overhead. Follows from IP-type-P Link Capacity [RFC5136].

bottleneck IP capacity: The IP capacity of the dominant bottleneck in the forward path. All throughput maximizing protocols estimate this capacity by observing the IP rate delivered through the bottleneck. Most protocols derive their self clocks from the timing of this data. See Section 4.1 and Appendix B for more details.

implied bottleneck IP capacity: This is the bottleneck IP capacity implied by the ACKs returning from the receiver. It is determined by looking at how much application data the ACK stream at the sender reports delivered to the data receiver per unit time at various time scales. If the return path is thinning, batching or otherwise altering the ACK timing the implied bottleneck IP capacity over short time scales might be substantially larger than the bottleneck IP capacity averaged over a full RTT. Since TCP derives its clock from the data delivered through the bottleneck the front path must have sufficient buffering to absorb any data bursts at the dimensions (duration and IP rate) implied by the ACK

stream, potentially doubled during slowstart. If the return path is not altering the ACK stream, then the implied bottleneck IP capacity will be the same as the bottleneck IP capacity. See Section 4.1 and Appendix B for more details.

sender interface rate: The IP rate which corresponds to the IP capacity of the data sender's interface. Due to sender efficiency algorithms including technologies such as TCP segmentation offload (TSO), nearly all moderns servers deliver data in bursts at full interface link rate. Today 1 or 10 Gb/s are typical.

Header_overhead: The IP and TCP header sizes, which are the portion of each MTU not available for carrying application payload. Without loss of generality this is assumed to be the size for returning acknowledgements (ACKs). For TCP, the Maximum Segment Size (MSS) is the Target MTU minus the header_overhead.

Basic parameters common to models and subpath tests are defined here are described in more detail in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and IP performance (which include TCP headers and retransmissions as part of the IP payload).

Window [size]: The total quantity of data plus the data represented by ACKs circulating in the network is referred to as the window. See Section 4.1. Sometimes used with other qualifiers (congestion window, cwnd or receiver window) to indicate which mechanism is controlling the window.

pipe size: A general term for number of packets needed in flight (the window size) to exactly fill some network path or subpath. It corresponds to the window size which maximizes network power, the observed data rate divided by the observed RTT. Often used with additional qualifiers to specify which path, or under what conditions, etc.

target_window_size: The average number of packets in flight (the window size) needed to meet the Target Data Rate, for the specified Target RTT, and MTU. It implies the scale of the bursts that the network might experience.

run length: A general term for the observed, measured, or specified number of packets that are (expected to be) delivered between losses or ECN Congestion Experienced (CE) marks. Nominally one over the sum of the loss and ECN CE marking probabilities, if there are independently and identically distributed.

target_run_length: The target_run_length is an estimate of the minimum number of non-congestion marked packets needed between losses or ECN Congestion Experienced (CE) marks necessary to attain the target_data_rate over a path with the specified target_RTT and target_MTU, as computed by a mathematical model of TCP congestion control. A reference calculation is shown in Section 5.2 and alternatives in Appendix A

reference target_run_length: target_run_length computed precisely by the method in Section 5.2. This is likely to be more slightly conservative than required by modern TCP implementations.

Ancillary parameters used for some tests:

derating: Under some conditions the standard models are too conservative. The modeling framework permits some latitude in relaxing or "derating" some test parameters as described in Section 5.3 in exchange for a more stringent TIDS validation procedures, described in Section 10.

subpath_IP_capacity: The IP capacity of a specific subpath.

test path: A subpath of a complete path under test.

test_path_RTT: The RTT observed between two measurement points using packet sizes that are consistent with the transport protocol. Generally MTU sized packets of the forward path, header_overhead sized packets on the return path.

test_path_pipe: The pipe size of a test path. Nominally the test path RTT times the test path IP_capacity.

test_window: The window necessary to meet the target_rate over a test path. Typically test_window=target_data_rate*test_path_RTT/ (target_MTU - header_overhead).

The terminology below is used to define temporal patterns for test stream. These patterns are designed to mimic TCP behavior, as described in Section 4.1.

packet headway: Time interval between packets, specified from the start of one to the start of the next. e.g. If packets are sent with a 1 mS headway, there will be exactly 1000 packets per second.

burst headway: Time interval between bursts, specified from the start of the first packet one burst to the start of the first packet of the next burst. e.g. If 4 packet bursts are sent with a 1 mS burst headway, there will be exactly 4000 packets per second.

paced single packets: Send individual packets at the specified rate or packet headway.

paced bursts: Send bursts on a timer. Specify any 3 of: average data rate, packet size, burst size (number of packets) and burst headway (burst start to start). By default the bursts are assumed full sender interface rate, such that the packet headway within each burst is the minimum supported by the sender's interface. Under some conditions it is useful to explicitly specify the packet headway within each burst.

slowstart rate: Mimic TCP slowstart by sending 4 packet paced bursts at an average data rate equal to twice the implied bottleneck IP capacity (but not more than the sender interface rate). This is a two level burst pattern described in more detail in Section 6.1. If the implied bottleneck IP capacity is more than half of the

sender interface rate, slowstart rate becomes sender interface rate.

slowstart burst: Mimic one round of TCP slowstart by sending a specified number of packets packets in a two level burst pattern that resembles slowstart.

repeated slowstart bursts: Repeat Slowstart bursts once per target_RTT. For TCP each burst would be twice as large as the prior burst, and the sequence would end at the first ECN CE mark or lost packet. For measurement, all slowstart bursts would be the same size (nominally target_window_size but other sizes might be specified), and the ECN CE marks and lost packets are counted.

The tests described in this note can be grouped according to their applicability.

Capacity tests: Capacity tests determine if a network subpath has sufficient capacity to deliver the Target Transport Performance. As long as the test stream is within the proper envelope for the Target Transport Performance, the average packet losses or ECN Congestion Experienced (CE) marks must be below the threshold computed by the model. As such, capacity tests reflect parameters that can transition from passing to failing as a consequence of cross traffic, additional presented load or the actions of other network users. By definition, capacity tests also consume significant network resources (data capacity and/or queue buffer space), and the test schedules must be balanced by their cost.

Monitoring tests: Monitoring tests are designed to capture the most important aspects of a capacity test, but without presenting excessive ongoing load themselves. As such they may miss some details of the network's performance, but can serve as a useful reduced-cost proxy for a capacity test, for example to support ongoing monitoring.

Engineering tests: Engineering tests evaluate how network algorithms (such as AQM and channel allocation) interact with TCP-style self clocked protocols and adaptive congestion control based on packet loss and ECN Congestion Experienced (CE) marks. These tests are likely to have complicated interactions with cross traffic and under some conditions can be inversely sensitive to load. For example a test to verify that an AQM algorithm causes ECN CE marks or packet drops early enough to limit queue occupancy may experience a false pass result in the presence of cross traffic. It is important that engineering tests be performed under a wide range of conditions, including both in situ and bench testing, and over a wide variety of load conditions. Ongoing monitoring is less likely to be useful for engineering tests, although sparse in situ testing might be appropriate.

4. Background

At the time the IPPM WG was chartered, sound Bulk Transport Capacity measurement was known to be well beyond our capabilities. Even at the time that Framework for IP Performance Metrics [RFC3148] was written we knew that we didn't fully understand the problem. Now, by hindsight we understand why BTC is such a hard problem:

- TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test.
- Congestion control is an equilibrium process, such that transport protocols change the packet transfer statistics (raise the packet loss ratio and/or RTT) to conform to their behavior. By design TCP congestion control keeps raising the data rate until the network gives some indication that it is full by dropping or ECN CE marking packets. If TCP successfully fills the network (e.g. the bottleneck is 100% utilized) the packet loss and ECN CE marks are mostly determined by TCP and how hard TCP drives the network at that rate and not by the network itself.
- TCP's ability to compensate for network flaws is directly proportional to the number of roundtrips per second (i.e. inversely proportional to the RTT). As a consequence a flawed subpath may pass a short RTT local test even though it fails when the subpath is extended by an effectively perfect network to some larger RTT.
- TCP has an extreme form of the Heisenberg problem - Measurement and cross traffic interact in unknown and ill defined ways. The situation is actually worse than the traditional physics problem where you can at least estimate bounds on the relative momentum of the measurement and measured particles. For network measurement you can not in general determine the relative "mass" of either the test stream or the cross traffic, so you can not gauge the relative magnitude of the uncertainty that might be introduced by any interaction.

These properties are a consequence of the equilibrium behavior intrinsic to how all throughput maximizing protocols interact with the Internet. These protocols rely on control systems based on estimated network parameters to regulate the quantity of data sent into the network. The sent data in turn alters the network properties observed by the estimators, such that there are circular dependencies between every component and every property. Since some of these properties are nonlinear, the entire system is nonlinear, and any change anywhere causes difficult to predict changes in every parameter.

Model Based Metrics overcome these problems by making the measurement system open loop: the packet transfer statistics (akin to the network

estimators) do not affect the traffic or traffic patterns (bursts), which computed on the basis of the Target Transport Performance. In order for a network to pass, the resulting packet transfer statistics and corresponding network estimators have to be such that they would not cause the control systems slow the traffic below the Target Data Rate.

4.1. TCP properties

TCP and SCTP are self clocked protocols that carry the vast majority of all Internet data. Their dominant behavior is to have an approximately fixed quantity of data and acknowledgements (ACKs) circulating in the network. The data receiver reports arriving data by returning ACKs to the data sender, the data sender typically responds by sending exactly the same quantity of data back into the network. The total quantity of data plus the data represented by ACKs circulating in the network is referred to as the window. The mandatory congestion control algorithms incrementally adjust the window by sending slightly more or less data in response to each ACK. The fundamentally important property of this system is that it is self clocked: The data transmissions are a reflection of the ACKs that were delivered by the network, the ACKs are a reflection of the data arriving from the network.

A number of protocol features cause bursts of data, even in idealized networks that can be modeled as simple queueing systems.

During slowstart the IP rate is doubled on each RTT by sending twice as much data as was delivered to the receiver during the prior RTT. Each returning ACK causes the sender to transmit twice the data the ACK reported arriving at the receiver. For slowstart to be able to fill a network the network must be able to tolerate slowstart bursts up to the full pipe size inflated by the anticipated window reduction on the first loss or ECN CE mark. For example, with classic Reno congestion control, an optimal slowstart has to end with a burst that is twice the bottleneck rate for one RTT in duration. This burst causes a queue which is equal to the pipe size (i.e. the window is twice the pipe size) so when the window is halved in response to the first packet loss, the new window will be the pipe size.

Note that if the bottleneck IP rate is less that half of the capacity of the front path (which is almost always the case), the slowstart bursts will not by themselves cause significant queues anywhere else along the front path; they primarily exercise the queue at the dominant bottleneck.

Several common efficiency algorithms also cause bursts. The self clock is typically applied to groups of packets: the receiver's

delayed ACK algorithm generally sends only one ACK per two data segments. Furthermore the modern senders use TCP segmentation offload (TSO) to reduce CPU overhead. The sender's software stack builds supersized TCP segments that the TSO hardware splits into MTU sized segments on the wire. The net effect of TSO, delayed ACK and other efficiency algorithms is to send bursts of segments at full sender interface rate.

Note that these efficiency algorithms are almost always in effect, including during slowstart, such that slowstart typically has a two level burst structure. Section 6.1 describes slowstart in more detail.

Additional sources of bursts include TCP's initial window [RFC6928], application pauses, channel allocation mechanisms and network devices that schedule ACKs. Appendix B describes these last two items. If the application pauses (stops reading or writing data) for some fraction of an RTT, many TCP implementations catch up to their earlier window size by sending a burst of data at the full sender interface rate. To fill a network with a realistic application, the network has to be able to tolerate sender interface rate bursts large enough to restore the prior window following application pauses.

Although the sender interface rate bursts are typically smaller than the last burst of a slowstart, they are at a higher IP rate so they potentially exercise queues at arbitrary points along the front path from the data sender up to and including the queue at the dominant bottleneck. There is no model for how frequent or what sizes of sender rate bursts the network should tolerate.

In conclusion, to verify that a path can meet a Target Transport Performance, it is necessary to independently confirm that the path can tolerate bursts at the scales that can be caused by these mechanisms. Three cases are believed to be sufficient:

- Two level slowstart bursts sufficient to get connections started properly.
- Ubiquitous sender interface rate bursts caused by efficiency algorithms. We assume 4 packet bursts to be the most common case, since it matches the effects of delayed ACK during slowstart. These bursts should be assumed not to significantly affect packet transfer statistics.
- Infrequent sender interface rate bursts that are full target_window_size. Target_run_length may be derated for these large fast bursts.

If a subpath can meet the required packet loss ratio for bursts at all of these scales then it has sufficient buffering at all potential

bottlenecks to tolerate any of the bursts that are likely introduced by TCP or other transport protocols.

4.2. Diagnostic Approach

A complete path of a given RTT and MTU, which are equal to or smaller than the Target RTT and equal to or larger than the Target MTU respectively, is expected to be able to attain a specified Bulk Transport Capacity when all of the following conditions are met:

1. The IP capacity is above the Target Data Rate by sufficient margin to cover all TCP/IP overheads. This can be confirmed by the tests described in Section 8.1 or any number of IP capacity tests adapted to implement MBM.
2. The observed packet transfer statistics are better than required by a suitable TCP performance model (e.g. fewer packet losses or ECN CE marks). See Section 8.1 or any number of low rate packet loss tests outside of MBM.
3. There is sufficient buffering at the dominant bottleneck to absorb a slowstart bursts large enough to get the flow out of slowstart at a suitable window size. See Section 8.3.
4. There is sufficient buffering in the front path to absorb and smooth sender interface rate bursts at all scales that are likely to be generated by the application, any channel arbitration in the ACK path or any other mechanisms. See Section 8.4.
5. When there is a slowly rising standing queue at the bottleneck the onset of packet loss has to be at an appropriate point (time or queue depth) and progressive [RFC7567]. See Section 8.2.
6. When there is a standing queue at a bottleneck for a shared media subpath (e.g. half duplex), there must be a suitable bounds on the interaction between ACKs and data, for example due to the channel arbitration mechanism. See Section 8.2.4.

Note that conditions 1 through 4 require capacity tests for validation, and thus may need to be monitored on an ongoing basis. Conditions 5 and 6 require engineering tests, which are best performed in controlled environments such as a bench test. They won't generally fail due to load, but may fail in the field due to configuration errors, etc. and should be spot checked.

We are developing a tool that can perform many of the tests described here [MBMSource].

4.3. New requirements relative to RFC 2330

Model Based Metrics are designed to fulfill some additional requirements that were not recognized at the time RFC 2330 was written [RFC2330]. These missing requirements may have significantly contributed to policy difficulties in the IP measurement space. Some

additional requirements are:

- o IP metrics must be actionable by the ISP - they have to be interpreted in terms of behaviors or properties at the IP or lower layers, that an ISP can test, repair and verify.
- o Metrics should be spatially composable, such that measures of concatenated paths should be predictable from subpaths.
- o Metrics must be vantage point invariant over a significant range of measurement point choices, including off path measurement points. The only requirements on MP selection should be that the RTT between the MPs is below some reasonable bound, and that the effects of the "test leads" connecting MPs to the subpath under test can be can be calibrated out of the measurements. The latter might be be accomplished if the test leads are effectively ideal or their properties can be deducted from the measurements between the MPs. While many of tests require that the test leads have at least as much IP capacity as the subpath under test, some do not, for example Background Packet Transfer Tests described in Section 8.1.3.
- o Metric measurements must be repeatable by multiple parties with no specialized access to MPs or diagnostic infrastructure. It must be possible for different parties to make the same measurement and observe the same results. In particular it is specifically important that both a consumer (or their delegate) and ISP be able to perform the same measurement and get the same result. Note that vantage independence is key to meeting this requirement.

5. Common Models and Parameters

5.1. Target End-to-end parameters

The target end-to-end parameters are the Target Data Rate, Target RTT and Target MTU as defined in Section 3. These parameters are determined by the needs of the application or the ultimate end user and the complete Internet path over which the application is expected to operate. The target parameters are in units that make sense to upper layers: payload bytes delivered to the application, above TCP. They exclude overheads associated with TCP and IP headers, retransmits and other protocols (e.g. DNS).

Other end-to-end parameters defined in Section 3 include the effective bottleneck data rate, the sender interface data rate and the TCP and IP header sizes.

The target_data_rate must be smaller than all subpath IP capacities by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance for fluctuations in TCP's actual data rate. Specifying a

target_data_rate with insufficient headroom is likely to result in brittle measurements having little predictive value.

Note that the target parameters can be specified for a hypothetical path, for example to construct TIDS designed for bench testing in the absence of a real application; or for a live in situ test of production infrastructure.

The number of concurrent connections is explicitly not a parameter to this model. If a subpath requires multiple connections in order to meet the specified performance, that must be stated explicitly and the procedure described in Section 6.4 applies.

5.2. Common Model Calculations

The Target Transport Performance is used to derive the target_window_size and the reference target_run_length.

The target_window_size, is the average window size in packets needed to meet the target_rate, for the specified target_RTT and target_MTU. It is given by:

target_window_size = ceiling(target_rate * target_RTT / (target_MTU - header_overhead))

Target_run_length is an estimate of the minimum required number of unmarked packets that must be delivered between losses or ECN Congestion Experienced (CE) marks, as computed by a mathematical model of TCP congestion control. The derivation here follows [MSMO97], and by design is quite conservative.

Reference target_run_length is derived as follows: assume the subpath_IP_capacity is infinitesimally larger than the target_data_rate plus the required header_overhead. Then target_window_size also predicts the onset of queueing. A larger window will cause a standing queue at the bottleneck.

Assume the transport protocol is using standard Reno style Additive Increase, Multiplicative Decrease (AIMD) congestion control [RFC5681] (but not Appropriate Byte Counting [RFC3465]) and the receiver is using standard delayed ACKs. Reno increases the window by one packet every pipe_size worth of ACKs. With delayed ACKs this takes 2 Round Trip Times per increase. To exactly fill the pipe, losses must be no closer than when the peak of the AIMD sawtooth reached exactly twice the target_window_size otherwise the multiplicative window reduction triggered by the loss would cause the network to be underfilled. Following [MSMO97] the number of packets between losses must be the area under the AIMD sawtooth. They must be no more frequent than

every 1 in ((3/2)*target_window_size)*(2*target_window_size) packets, which simplifies to:

target_run_length = 3*(target_window_size^2)

Note that this calculation is very conservative and is based on a number of assumptions that may not apply. Appendix A discusses these assumptions and provides some alternative models. If a different model is used, a fully specified TIDS or FSTIDS MUST document the actual method for computing target_run_length and ratio between alternate target_run_length and the reference target_run_length calculated above, along with a discussion of the rationale for the underlying assumptions.

These two parameters, target_window_size and target_run_length, directly imply most of the individual parameters for the tests in Section 8.

5.3. Parameter Derating

Since some aspects of the models are very conservative, the MBM framework permits some latitude in derating test parameters. Rather than trying to formalize more complicated models we permit some test parameters to be relaxed as long as they meet some additional procedural constraints:

- The TIDS or FSTIDS MUST document and justify the actual method used to compute the derated metric parameters.
- The validation procedures described in Section 10 must be used to demonstrate the feasibility of meeting the Target Transport Performance with infrastructure that infinitesimally passes the derated tests.
- The validation process for a FSTIDS itself must be documented is such a way that other researchers can duplicate the validation experiments.

Except as noted, all tests below assume no derating. Tests where there is not currently a well established model for the required parameters explicitly include derating as a way to indicate flexibility in the parameters.

5.4. Test Preconditions

Many tests have preconditions which are required to assure their validity. Examples include: the presence or nonpresence of cross traffic on specific subpaths; negotiating ECN; and appropriate preloading to put reactive network elements into the proper states [RFC7312]. If preconditions are not properly satisfied for some reason, the tests should be considered to be inconclusive. In

general it is useful to preserve diagnostic information as to why the preconditions were not met, and any test data that was collected even if it is not useful for the intended test. Such diagnostic information and partial test data may be useful for improving the test in the future.

It is important to preserve the record that a test was scheduled, because otherwise precondition enforcement mechanisms can introduce sampling bias. For example, canceling tests due to cross traffic on subscriber access links might introduce sampling bias in tests of the rest of the network by reducing the number of tests during peak network load.

Test preconditions and failure actions MUST be specified in a FSTIDS.

6. Generating test streams

Many important properties of Model Based Metrics, such as vantage independence, are a consequence of using test streams that have temporal structures that mimic TCP or other transport protocols running over a complete path. As described in Section 4.1, self clocked protocols naturally have burst structures related to the RTT and pipe size of the complete path. These bursts naturally get larger (contain more packets) as either the Target RTT or Target Data Rate get larger, or the Target MTU gets smaller. An implication of these relationships is that test streams generated by running self clocked protocols over short subpaths may not adequately exercise the queueing at any bottleneck to determine if the subpath can support the full Target Transport Performance over the complete path.

Failing to authentically mimic TCP's temporal structure is part the reason why simple performance tools such as iperf, netperf, nc, etc have the reputation of yielding false pass results over short test paths, even when some subpath has a flaw.

The definitions in Section 3 are sufficient for most test streams. We describe the slowstart and standing queue test streams in more detail.

In conventional measurement practice stochastic processes are used to eliminate many unintended correlations and sample biases. However MBM tests are designed to explicitly mimic temporal correlations caused by network or protocol elements themselves and are intended to accurately reflect implementation behavior. Some portion of the system, such as traffic arrival (test scheduling) are naturally stochastic. Other details, such as protocol processing times, are technically nondeterministic and might be modeled stochastically, but

are only a tiny part of the overall behavior which is dominated by implementation specific deterministic effects. Furthermore, it is known that sampling bias is a real problem for some protocol implementations. For example TCP's RTT estimator used to determine the Retransmit Time Out (RTO), can be fooled by periodic cross traffic or start-stop applications.

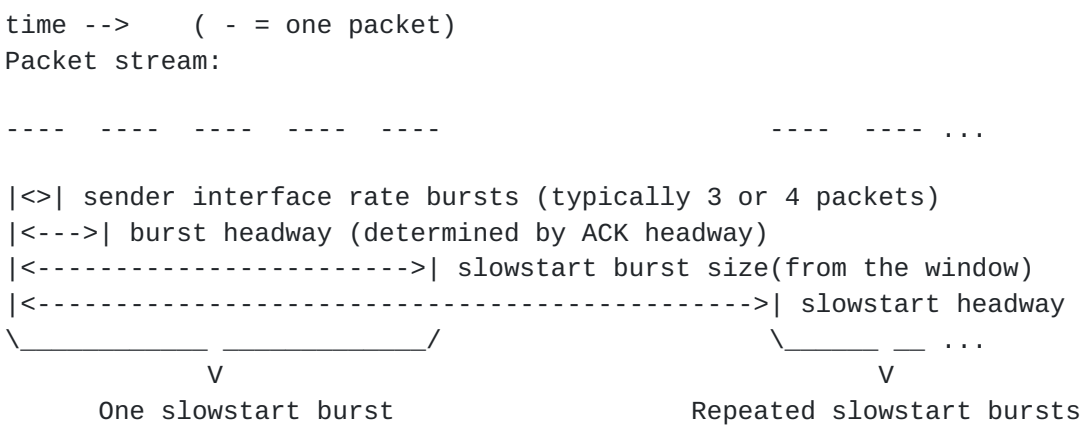
At some point in the future it may make sense to introduce fine grained noise sources into the models used for generating test streams, but they are not warranted at this time.

6.1. Mimicking slowstart

TCP slowstart has a two level burst structure as shown in Figure 2. The fine structure is caused by the interaction between the ACK clock and TCP efficiency algorithms. Each ACK passing through the return path triggers a small data burst. These bursts are typically full sender interface rate, with the same headway as the returning ACKs, but having twice as much data as the ACK reported was delivered to the receiver. Due to variations in delayed ACK and algorithms such as Appropriate Byte Counting [RFC3465], different pairs of senders and receivers produce different burst patterns. Without loss of generality, we assume each ACK causes 4 packet bursts at an average headway equal to the ACK headway, and corresponding to sending at an average rate equal to twice the effective bottleneck IP rate. This fine structure defines one slowstart rate burst.

For a transport protocol the slowstart bursts are repeated every target_RTT. Each slowstart burst is twice as large as the previous burst, and slowstart ends on the first lost packet or ECN mark. For diagnostic tests described below we preserve the fine structure but manipulate the burst size and headway to measure the ability of the dominant bottleneck to absorb and smooth slowstart bursts.

Note that a stream of repeated slowstart bursts has three different average rates, depending on the averaging interval. At the finest time scale (a few packet times at the sender interface) the peak of the average IP rate is the same as the sender interface rate; at a medium timescale (a few packet times at the dominant bottleneck) the peak of the average IP rate is twice the implied bottleneck IP capacity; and at time scales longer than the target_RTT and when the burst size is equal to the target_window_size the average rate is equal to the target_data_rate. This pattern corresponds to repeating the last RTT of TCP slowstart when delayed ACK and sender side byte counting are present but without the limits specified in Appropriate Byte Counting [RFC3465].



Multiple levels of Slowstart Bursts

Figure 2

6.2. Constant window pseudo CBR

Implement pseudo constant bit rate by running a standard protocol such as TCP with a fixed window size, such that it is self clocked. Data packets arriving at the receiver trigger acknowledgements (ACKs) which travel back to the sender where they trigger additional transmissions. The window size is computed from the target_data_rate and the actual RTT of the test path. The rate is only maintained in average over each RTT, and is subject to limitations of the transport protocol.

Since the window size is constrained to be an integer number of packets, for small RTTs or low data rates there may not be sufficiently precise control over the data rate. Rounding the window size up (the default) is likely to be result in data rates that are higher than the target rate, but reducing the window by one packet may result in data rates that are too small. Also cross traffic potentially raises the RTT, implicitly reducing the rate. Cross traffic that raises the RTT nearly always makes the test more strenuous. A FSTIDS specifying a constant window CBR tests MUST explicitly indicate under what conditions errors in the data rate causes tests to inconclusive.

Since constant window pseudo CBR testing is sensitive to RTT fluctuations it is less accurate at controling the data rate in environments with fluctuating delays.

6.3. Scanned window pseudo CBR

Scanned window pseudo CBR is similar to the constant window CBR described above, except the window is scanned across a range of sizes

designed to include two key events, the onset of queueing and the onset of packet loss or ECN CE marks. The window is scanned by incrementing it by one packet every 2*target_window_size delivered packets. This mimics the additive increase phase of standard Reno TCP congestion avoidance when delayed ACKs are in effect. Normally the window increases separated by intervals slightly longer than twice the target_RTT.

There are two ways to implement this test: one built by applying a window clamp to standard congestion control in a standard protocol such as TCP and the other built by stiffening a non-standard transport protocol. When standard congestion control is in effect, any losses or ECN CE marks cause the transport to revert to a window smaller than the clamp such that the scanning clamp loses control the window size. The NPAD pathdiag tool is an example of this class of algorithms [Pathdiag].

Alternatively a non-standard congestion control algorithm can respond to losses by transmitting extra data, such that it maintains the specified window size independent of losses or ECN CE marks. Such a stiffened transport explicitly violates mandatory Internet congestion control [RFC5681] and is not suitable for in situ testing. It is only appropriate for engineering testing under laboratory conditions. The Windowed Ping tool implements such a test [WPING]. The tool described in the paper has been updated.[mpingSource]

The test procedures in Section 8.2 describe how to the partition the scans into regions and how to interpret the results.

6.4. Concurrent or channelized testing

The procedures described in this document are only directly applicable to single stream measurement, e.g. one TCP connection or measurement stream. In an ideal world, we would disallow all performance claims based multiple concurrent streams, but this is not practical due to at least two different issues. First, many very high rate link technologies are channelized and at last partially pin the flow to channel mapping to minimize packet reordering within flows. Second, TCP itself has scaling limits. Although the former problem might be overcome through different design decisions, the later problem is more deeply rooted.

All congestion control algorithms that are philosophically aligned with the standard [RFC5681] (e.g. claim some level of TCP compatibility, friendliness or fairness) have scaling limits, in the sense that as a long fast network (LFN) with a fixed RTT and MTU gets faster, these congestion control algorithms get less accurate and as a consequence have difficulty filling the network[CCscaling]. These

properties are a consequence of the original Reno AIMD congestion control design and the requirement in [RFC5681] that all transport protocols have similar responses to congestion.

There are a number of reasons to want to specify performance in term of multiple concurrent flows, however this approach is not recommended for data rates below several megabits per second, which can be attained with run lengths under 10000 packets on many paths. Since the required run length goes as the square of the data rate, at higher rates the run lengths can be unreasonably large, and multiple flows might be the only feasible approach.

If multiple flows are deemed necessary to meet aggregate performance targets then this MUST be stated both the design of the TIDS and in any claims about network performance. The IP diagnostic tests MUST be performed concurrently with the specified number of connections. For the the tests that use bursty test streams, the bursts should be synchronized across streams.

7. Interpreting the Results

7.1. Test outcomes

To perform an exhaustive test of a complete network path, each test of the TIDS is applied to each subpath of the complete path. If any subpath fails any test then a standard transport protocol running over the complete path can also be expected to fail to attain the Target Transport Performance under some conditions.

In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons. Proper instrumentation and treatment of inconclusive outcomes is critical to the accuracy and robustness of Model Based Metrics. Tests can be inconclusive if the precomputed traffic pattern or data rates were not accurately generated; the measurement results were not statistically significant; and others causes such as failing to meet some required preconditions for the test. See Section 5.4

For example consider a test that implements Constant Window Pseudo CBR (Section 6.2) by adding rate controls and detailed IP packet transfer instrumentation to TCP (e.g. [RFC4898]). TCP includes built in control systems which might interfere with the sending data rate. If such a test meets the required packet transfer statistics (e.g. run length) while failing to attain the specified data rate it must be treated as an inconclusive result, because we can not a priori determine if the reduced data rate was caused by a TCP problem or a network problem, or if the reduced data rate had a material

effect on the observed packet transfer statistics.

Note that for capacity tests, if the observed packet transfer statistics meet the statistical criteria for failing (accepting hypnosis H1 in Section 7.2), the test can can be considered to have failed because it doesn't really matter that the test didn't attain the required data rate.

The really important new properties of MBM, such as vantage independence, are a direct consequence of opening the control loops in the protocols, such that the test stream does not depend on network conditions or IP packets received. Any mechanism that introduces feedback between the path's measurements and the test stream generation is at risk of introducing nonlinearities that spoil these properties. Any exceptional event that indicates that such feedback has happened should cause the test to be considered inconclusive.

One way to view inconclusive tests is that they reflect situations where a test outcome is ambiguous between limitations of the network and some unknown limitation of the IP diagnostic test itself, which may have been caused by some uncontrolled feedback from the network.

Note that procedures that attempt to sweep the target parameter space to find the limits on some parameter such as target_data_rate are at risk of breaking the location independent properties of Model Based Metrics, if any part of the boundary between passing and inconclusive results is sensitive to RTT (which is normally the case).

One of the goals for evolving TIDS designs will be to keep sharpening distinction between inconclusive, passing and failing tests. The criteria for for passing, failing and inconclusive tests MUST be explicitly stated for every test in the TIDS or FSTIDS.

One of the goals of evolving the testing process, procedures, tools and measurement point selection should be to minimize the number of inconclusive tests.

It may be useful to keep raw packet transfer statistics and ancillary metrics [RFC3148] for deeper study of the behavior of the network path and to measure the tools themselves. Raw packet transfer statistics can help to drive tool evolution. Under some conditions it might be possible to reevaluate the raw data for satisfying alternate Target Transport Performance. However it is important to guard against sampling bias and other implicit feedback which can cause false results and exhibit measurement point vantage sensitivity. Simply applying different delivery criteria based on a different Target Transport Performance is insufficient if the test

traffic patterns (bursts, etc) does not match the alternate Target Transport Performance.

7.2. Statistical criteria for estimating run_length

When evaluating the observed run_length, we need to determine appropriate packet stream sizes and acceptable error levels for efficient measurement. In practice, can we compare the empirically estimated packet loss and ECN Congestion Experienced (CE) marking ratios with the targets as the sample size grows? How large a sample is needed to say that the measurements of packet transfer indicate a particular run length is present?

The generalized measurement can be described as recursive testing: send packets (individually or in patterns) and observe the packet delivery performance (packet loss ratio or other metric, any marking we define).

As each packet is sent and measured, we have an ongoing estimate of the performance in terms of the ratio of packet loss or ECN CE mark to total packets (i.e. an empirical probability). We continue to send until conditions support a conclusion or a maximum sending limit has been reached.

We have a target_mark_probability, 1 mark per target_run_length, where a "mark" is defined as a lost packet, a packet with ECN CE mark, or other signal. This constitutes the null Hypothesis:

H0: no more than one mark in target_run_length =
3*(target_window_size)^2 packets

and we can stop sending packets if on-going measurements support accepting H0 with the specified Type I error = alpha (= 0.05 for example).

We also have an alternative Hypothesis to evaluate: if performance is significantly lower than the target_mark_probability. Based on analysis of typical values and practical limits on measurement duration, we choose four times the H0 probability:

H1: one or more marks in (target_run_length/4) packets

and we can stop sending packets if measurements support rejecting H0 with the specified Type II error = beta (= 0.05 for example), thus preferring the alternate hypothesis H1.

H0 and H1 constitute the Success and Failure outcomes described elsewhere in the memo, and while the ongoing measurements do not

support either hypothesis the current status of measurements is inconclusive.

The problem above is formulated to match the Sequential Probability Ratio Test (SPRT) [StatQC]. Note that as originally framed the events under consideration were all manufacturing defects. In networking, ECN CE marks and lost packets are not defects but signals, indicating that the transport protocol should slow down.

The Sequential Probability Ratio Test also starts with a pair of hypothesis specified as above:

H0: p0 = one defect in target_run_length
H1: p1 = one defect in target_run_length/4
As packets are sent and measurements collected, the tester evaluates the cumulative defect count against two boundaries representing H0 Acceptance or Rejection (and acceptance of H1):

Acceptance line: Xa = -h1 + s*n
Rejection line: Xr = h2 + s*n
where n increases linearly for each packet sent and

h1 = { log((1-alpha)/beta) }/k
h2 = { log((1-beta)/alpha) }/k
k = log{ (p1(1-p0)) / (p0(1-p1)) }
s = [log{ (1-p0)/(1-p1) }]/k
for p0 and p1 as defined in the null and alternative Hypotheses statements above, and alpha and beta as the Type I and Type II errors.

The SPRT specifies simple stopping rules:

- Xa < defect_count(n) < Xb: continue testing
- defect_count(n) <= Xa: Accept H0
- defect_count(n) >= Xb: Accept H1

The calculations above are implemented in the R-tool for Statistical Analysis [Rtool] , in the add-on package for Cross-Validation via Sequential Testing (CVST) [CVST] .

Using the equations above, we can calculate the minimum number of packets (n) needed to accept H0 when x defects are observed. For example, when x = 0:

```
Xa = 0  = -h1 + s*n
and  n = h1 / s
```

7.3. Reordering Tolerance

All tests must be instrumented for packet level reordering [RFC4737]. However, there is no consensus for how much reordering should be acceptable. Over the last two decades the general trend has been to make protocols and applications more tolerant to reordering (see for example [RFC4015]), in response to the gradual increase in reordering in the network. This increase has been due to the deployment of technologies such as multi threaded routing lookups and Equal Cost MultiPath (ECMP) routing. These techniques increase parallelism in network and are critical to enabling overall Internet growth to exceed Moore's Law.

Note that transport retransmission strategies can trade off reordering tolerance vs how quickly they can repair losses vs overhead from spurious retransmissions. In advance of new retransmission strategies we propose the following strawman: Transport protocols should be able to adapt to reordering as long as the reordering extent is not more than the maximum of one quarter window or 1 mS, whichever is larger. Within this limit on reorder extent, there should be no bound on reordering density.

By implication, recording which is less than these bounds should not be treated as a network impairment. However [RFC4737] still applies: reordering should be instrumented and the maximum reordering that can be properly characterized by the test (e.g. bound on history buffers) should be recorded with the measurement results.

Reordering tolerance and diagnostic limitations, such as the size of the history buffer used to diagnose packets that are way out-of-order, MUST be specified in a FSTIDS.

8. IP Diagnostic Tests

The IP diagnostic tests below are organized by traffic pattern: basic data rate and packet transfer statistics, standing queues, slowstart bursts, and sender rate bursts. We also introduce some combined tests which are more efficient when networks are expected to pass, but conflate diagnostic signatures when they fail.

There are a number of test details which are not fully defined here. They must be fully specified in a FSTIDS. From a standardization perspective, this lack of specificity will weaken this version of Model Based Metrics, however it is anticipated that this it be more

than offset by the extent to which MBM suppresses the problems caused by using transport protocols for measurement. e.g. non-specific MBM metrics are likely to have better repeatability than many existing BTC like metrics. Once we have good field experience, the missing details can be fully specified.

8.1. Basic Data Rate and Packet Transfer Tests

We propose several versions of the basic data rate and packet transfer statistics test. All measure the number of packets delivered between losses or ECN Congestion Experienced (CE) marks, using a data stream that is rate controlled at or below the target_data_rate.

The tests below differ in how the data rate is controlled. The data can be paced on a timer, or window controlled at full Target Data Rate. The first two tests implicitly confirm that sub_path has sufficient raw capacity to carry the target_data_rate. They are recommend for relatively infrequent testing, such as an installation or periodic auditing process. The third, background packet transfer statistics, is a low rate test designed for ongoing monitoring for changes in subpath quality.

All rely on the data receiver accumulating packet transfer statistics as described in Section 7.2 to score the outcome:

Pass: it is statistically significant that the observed interval between losses or ECN CE marks is larger than the target_run_length.

Fail: it is statistically significant that the observed interval between losses or ECN CE marks is smaller than the target_run_length.

A test is considered to be inconclusive if it failed to meet the data rate as specified below, meet the qualifications defined in Section 5.4 or neither run length statistical hypothesis was confirmed in the allotted test duration.

8.1.1. Delivery Statistics at Paced Full Data Rate

Confirm that the observed run length is at least the target_run_length while relying on timer to send data at the target_rate using the procedure described in in Section 6.1 with a burst size of 1 (single packets) or 2 (packet pairs).

The test is considered to be inconclusive if the packet transmission can not be accurately controlled for any reason.

RFC 6673 [RFC6673] is appropriate for measuring packet transfer

statistics at full data rate.

8.1.2. Delivery Statistics at Full Data Windowed Rate

Confirm that the observed run length is at least the target_run_length while sending at an average rate approximately equal to the target_data_rate, by controlling (or clamping) the window size of a conventional transport protocol to a fixed value computed from the properties of the test path, typically test_window=target_data_rate*test_path_RTT/target_MTU. Note that if there is any interaction between the forward and return path, test_window may need to be adjusted slightly to compensate for the resulting inflated RTT. However see the discussion in Section 8.2.4.

Since losses and ECN CE marks cause transport protocols to reduce their data rates, this test is expected to be less precise about controlling its data rate. It should not be considered inconclusive as long as at least some of the round trips reached the full target_data_rate without incurring losses or ECN CE marks. To pass this test the network MUST deliver target_window_size packets in target_RTT time without any losses or ECN CE marks at least once per two target_window_size round trips, in addition to meeting the run length statistical test.

8.1.3. Background Packet Transfer Statistics Tests

The background run length is a low rate version of the target target rate test above, designed for ongoing lightweight monitoring for changes in the observed subpath run length without disrupting users. It should be used in conjunction with one of the above full rate tests because it does not confirm that the subpath can support raw data rate.

RFC 6673 [RFC6673] is appropriate for measuring background packet transfer statistics.

8.2. Standing Queue Tests

These engineering tests confirm that the bottleneck is well behaved across the onset of packet loss, which typically follows after the onset of queueing. Well behaved generally means lossless for transient queues, but once the queue has been sustained for a sufficient period of time (or reaches a sufficient queue depth) there should be a small number of losses to signal to the transport protocol that it should reduce its window. Losses that are too early can prevent the transport from averaging at the target_data_rate. Losses that are too late indicate that the queue might be subject to bufferbloat [wikiBloat] and inflict excess queuing delays on all

flows sharing the bottleneck queue. Excess losses (more than half of the window) at the onset of congestion make loss recovery problematic for the transport protocol. Non-linear, erratic or excessive RTT increases suggest poor interactions between the channel acquisition algorithms and the transport self clock. All of the tests in this section use the same basic scanning algorithm, described here, but score the link or subpath on the basis of how well it avoids each of these problems.

For some technologies the data might not be subject to increasing delays, in which case the data rate will vary with the window size all the way up to the onset of load induced packet loss or ECN CE marks. For theses technologies, the discussion of queueing does not apply, but it is still required that the onset of losses or ECN CE marks be at an appropriate point and progressive.

Use the procedure in Section 6.3 to sweep the window across the onset of queueing and the onset of loss. The tests below all assume that the scan emulates standard additive increase and delayed ACK by incrementing the window by one packet for every 2*target_window_size packets delivered. A scan can typically be divided into three regions: below the onset of queueing, a standing queue, and at or beyond the onset of loss.

Below the onset of queueing the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the subpath IP rate, the data rate becomes fairly constant, and the RTT increases in proportion to the increase in window size. The precise transition across the start of queueing can be identified by the maximum network power, defined to be the ratio data rate over the RTT. The network power can be computed at each window size, and the window with the maximum are taken as the start of the queueing region.

For technologies that do not have conventional queues, start the scan at a window equal to the test_window=target_data_rate*test_path_RTT/ target_MTU, i.e. starting at the target rate, instead of the power point.

If there is random background loss (e.g. bit errors, etc), precise determination of the onset of queue induced packet loss may require multiple scans. Above the onset of queuing loss, all transport protocols are expected to experience periodic losses determined by the interaction between the congestion control and AQM algorithms. For standard congestion control algorithms the periodic losses are likely to be relatively widely spaced and the details are typically dominated by the behavior of the transport protocol itself. For the stiffened transport protocols case (with non-standard, aggressive

congestion control algorithms) the details of periodic losses will be dominated by how the the window increase function responds to loss.

8.2.1. Congestion Avoidance

A subpath passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between the onset of queueing (as determined by the window with the maximum network power) and the first loss or ECN CE mark. If this test is implemented using a standards congestion control algorithm with a clamp, it can be performed in situ in the production internet as a capacity test. For an example of such a test see [Pathdiag].

For technologies that do not have conventional queues, use the test_window in place of the onset of queueing. i.e. A subpath passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between start of the scan at test_window and the first loss or ECN CE mark.

8.2.2. Bufferbloat

This test confirms that there is some mechanism to limit buffer occupancy (e.g. that prevents bufferbloat). Note that this is not strictly a requirement for single stream bulk transport capacity, however if there is no mechanism to limit buffer queue occupancy then a single stream with sufficient data to deliver is likely to cause the problems described in [RFC7567], and [wikiBloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the subpath unusable for other flows and applications.

Pass if the onset of loss occurs before a standing queue has introduced more delay than than twice target_RTT, or other well defined and specified limit. Note that there is not yet a model for how much standing queue is acceptable. The factor of two chosen here reflects a rule of thumb. In conjunction with the previous test, this test implies that the first loss should occur at a queueing delay which is between one and two times the target_RTT.

Specified RTT limits that are larger than twice the target_RTT must be fully justified in the FSTIDS.

8.2.3. Non excessive loss

This test confirm that the onset of loss is not excessive. Pass if losses are equal or less than the increase in the cross traffic plus the test stream window increase on the previous RTT. This could be restated as non-decreasing subpath throughput at the onset of loss, which is easy to meet as long as discarding packets is not more

expensive than delivering them. (Note when there is a transient drop in subpath throughput, outside of a standing queue test, a subpath that passes other queue tests in this document will have sufficient queue space to hold one RTT worth of data).

Note that conventional Internet policers will not pass this test, which is correct. TCP often fails to come into equilibrium at more than a small fraction of the available capacity, if the capacity is enforced by a policer. [Citation Pending].

8.2.4. Duplex Self Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path.

Some historical half duplex technologies had the property that each direction held the channel until it completely drained its queue. When a self clocked transport protocol, such as TCP, has data and ACKs passing in opposite directions through such a link, the behavior often reverts to stop-and-wait. Each additional packet added to the window raises the observed RTT by two packet times, once as it passes through the data path, and once for the additional delay incurred by the ACK waiting on the return path.

The duplex self interference test fails if the RTT rises by more than a fixed bound above the expected queueing time computed from trom the excess window divided by the subpath IP Capacity. This bound must be smaller than target_RTT/2 to avoid reverting to stop and wait behavior. (e.g. Data packets and ACKs both have to be released at least twice per RTT.)

8.3. Slowstart tests

These tests mimic slowstart: data is sent at twice the effective bottleneck rate to exercise the queue at the dominant bottleneck.

8.3.1. Full Window slowstart test

This is a capacity test to confirm that slowstart is not likely to exit prematurely. Send slowstart bursts that are target_window_size total packets.

Accumulate packet transfer statistics as described in Section 7.2 to score the outcome. Pass if it is statistically significant that the observed number of good packets delivered between losses or ECN CE marks is larger than the target_run_length. Fail if it is statistically significant that the observed interval between losses or ECN CE marks is smaller than the target_run_length.

It is deemed inconclusive if the elapsed time to send the data burst is not less than half of the time to receive the ACKs. (i.e. sending data too fast is ok, but sending it slower than twice the actual bottleneck rate as indicated by the ACKs is deemed inconclusive). The headway for the slowstart bursts should be the target_RTT.

Note that these are the same parameters as the Sender Full Window burst test, except the burst rate is at slowestart rate, rather than sender interface rate.

8.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at twice the implied IP bottleneck capacity), until the first loss, stop, allow the network to drain and repeat, gathering statistics on how many packets were delivered before the loss, the pattern of losses, maximum observed RTT and window size. Justify the results. There is not currently sufficient theory justifying requiring any particular result, however design decisions that affect the outcome of this tests also affect how the network balances between long and short flows (the "mice vs elephants" problem). The queue at the time of the first loss should be at least one half of the target_RTT.

This is an engineering test: It must be performed on a quiescent network or testbed, since cross traffic has the potential to change the results.

8.4. Sender Rate Burst tests

These tests determine how well the network can deliver bursts sent at sender's interface rate. Note that this test most heavily exercises the front path, and is likely to include infrastructure may be out of scope for an access ISP, even though the bursts might be caused by ACK compression, thinning or channel arbitration in the access ISP. See Appendix B.

Also, there are a several details that are not precisely defined. For starters there is not a standard server interface rate. 1 Gb/s and 10 Gb/s are common today, but higher rates will become cost effective and can be expected to be dominant some time in the future.

Current standards permit TCP to send a full window bursts following an application pause. (Congestion Window Validation [RFC2861], is not required, but even if was, it does not take effect until an application pause is longer than an RTO.) Since full window bursts are consistent with standard behavior, it is desirable that the network be able to deliver such bursts, otherwise application pauses will cause unwarranted losses. Note that the AIMD sawtooth requires

a peak window that is twice target_window_size, so the worst case burst may be 2*target_window_size.

It is also understood in the application and serving community that interface rate bursts have a cost to the network that has to be balanced against other costs in the servers themselves. For example TCP Segmentation Offload (TSO) reduces server CPU in exchange for larger network bursts, which increase the stress on network buffer memory. Some newer TCP implementations can pace traffic at scale [TSO_pacing][TSO_fq_pacing]. It remains to be determined if and how quickly these changes will be deployed.

There is not yet theory to unify these costs or to provide a framework for trying to optimize global efficiency. We do not yet have a model for how much the network should tolerate server rate bursts. Some bursts must be tolerated by the network, but it is probably unreasonable to expect the network to be able to efficiently deliver all data as a series of bursts.

For this reason, this is the only test for which we encourage derating. A TIDS could include a table of pairs of derating parameters: burst sizes and how much each burst size is permitted to reduce the run length, relative to to the target_run_length.

8.5. Combined and Implicit Tests

Combined tests efficiently confirm multiple network properties in a single test, possibly as a side effect of normal content delivery. They require less measurement traffic than other testing strategies at the cost of conflating diagnostic signatures when they fail. These are by far the most efficient for monitoring networks that are nominally expected to pass all tests.

8.5.1. Sustained Bursts Test

The sustained burst test implements a combined worst case version of all of the capacity tests above. It is simply:

Send target_window_size bursts of packets at server interface rate with target_RTT burst headway (burst start to burst start). Verify that the observed packet transfer statistics meets the target_run_length.

Key observations:

- The subpath under test is expected to go idle for some fraction of the time: (subpath_IP_capacity-target_rate/ (target_MTU-header_overhead)*target_MTU)/subpath_IP_capacity. Failing to do so indicates a problem with the procedure and an

inconclusive test result.
- The burst sensitivity can be derated by sending smaller bursts more frequently. E.g. send target_window_size*derate packet bursts every target_RTT*derate.
- When not derated, this test is the most strenuous capacity test.
- A subpath that passes this test is likely to be able to sustain higher rates (close to subpath_IP_capacity) for paths with RTTs significantly smaller than the target_RTT.
- This test can be implemented with instrumented TCP [RFC4898], using a specialized measurement application at one end [MBMSource] and a minimal service at the other end [RFC0863] [RFC0864].
- This test is efficient to implement, since it does not require per-packet timers, and can make use of TSO in modern NIC hardware.
- If a subpath is known to pass the Standing Queue engineering tests (particularly that it has a progressive onset of loss at an appropriate queue depth), then the Sustained Burst Test is sufficient to assure that the subpath under test will not impair Bulk Transport Capacity at the target performance under all conditions. See Section 8.2 for a discussion of the standing queue tests.

Note that this test is clearly independent of the subpath RTT, or other details of the measurement infrastructure, as long as the measurement infrastructure can accurately and reliably deliver the required bursts to the subpath under test.

8.5.2. Streaming Media

Model Based Metrics can be implicitly implemented as a side effect any non-throughput maximizing application, such as streaming media, with some additional controls and instrumentation in the servers. The essential requirement is that the data rate be constrained such that even with arbitrary application pauses and bursts the data rate and burst sizes stay within the envelope defined by the individual tests described above.

If the application's serving_data_rate is less than or equal to the target_data_rate and the serving_RTT (the RTT between the sender and client) is less than the target_RTT, this constraint is most easily implemented by clamping the transport window size to be no larger than:

serving_window_clamp=target_data_rate*serving_RTT/
(target_MTU-header_overhead)

Under the above constraints the serving_window_clamp will limit the both the serving data rate and burst sizes to be no larger than the procedures in Section 8.1.2 and Section 8.4 or Section 8.5.1. Since

the serving RTT is smaller than the target_RTT, the worst case bursts that might be generated under these conditions will be smaller than called for by Section 8.4 and the sender rate burst sizes are implicitly derated by the serving_window_clamp divided by the target_window_size at the very least. (Depending on the application behavior, the data might be significantly smoother than specified by any of the burst tests.)

In an alternative implementation the data rate and bursts might be explicitly controlled by a programmable traffic shaper or pacing at the sender. This would provide better control over transmissions but it is substantially more complicated to implement and would be likely to have a higher CPU overhead.

Note that these techniques can be applied to any content delivery that can be subjected to a reduced data rate in order to inhibit TCP equilibrium behavior.

9. An Example

In this section a we illustrate a TIDS designed to confirm that an access ISP can reliably deliver HD video from multiple content providers to all of their customers. With modern codecs, minimal HD video (720p) generally fits in 2.5 Mb/s. Due to their geographical size, network topology and modem designs the ISP determines that most content is within a 50 mS RTT from their users (This is a sufficient to cover continental Europe or either US coast from a single serving site.)

2.5 Mb/s over a 50 ms path

End-to-End Parameter	value	units
target_rate	2.5	Mb/s
target_RTT	50	ms
target_MTU	1500	bytes
header_overhead	64	bytes
target_window_size	11	packets
target_run_length	363	packets

Table 1

Table 1 shows the default TCP model with no derating, and as such is quite conservative. The simplest TIDS would be to use the sustained burst test, described in Section 8.5.1. Such a test would send 11

packet bursts every 50mS, and confirming that there was no more than 1 packet loss per 33 bursts (363 total packets in 1.650 seconds).

Since this number represents is the entire end-to-end loss budget, independent subpath tests could be implemented by apportioning the packet loss ratio across subpaths. For example 50% of the losses might be allocated to the access or last mile link to the user, 40% to the interconnects with other ISPs and 1% to each internal hop (assuming no more than 10 internal hops). Then all of the subpaths can be tested independently, and the spatial composition of passing subpaths would be expected to be within the end-to-end loss budget.

Testing interconnects has generally been problematic: conventional performance tests run between measurement points adjacent to either side of the interconnect, are not generally useful. Unconstrained TCP tests, such as iperf [iperf] are usually overly aggressive because the RTT is so small (often less than 1 mS). With a short RTT these tools are likely to report inflated numbers because for short RTTs these tools can tolerate very high packet loss ratios and can push other cross traffic off of the network. As a consequence they are useless for predicting actual user performance, and may themselves be quite disruptive. Model Based Metrics solves this problem. The same test pattern as used on other subpaths can be applied to the interconnect. For our example, when apportioned 40% of the losses, 11 packet bursts sent every 50mS should have fewer than one loss per 82 bursts (902 packets).

10. Validation

Since some aspects of the models are likely to be too conservative, Section 5.2 permits alternate protocol models and Section 5.3 permits test parameter derating. If either of these techniques are used, we require demonstrations that such a TIDS can robustly detect subpaths that will prevent authentic applications using state-of-the-art protocol implementations from meeting the specified Target Transport Performance. This correctness criteria is potentially difficult to prove, because it implicitly requires validating a TIDS against all possible subpaths and subpaths. The procedures described here are still experimental.

We suggest two approaches, both of which should be applied: first, publish a fully open description of the TIDS, including what assumptions were used and and how it was derived, such that the research community can evaluate the design decisions, test them and comment on their applicability; and second, demonstrate that an applications running over an infinitesimally passing testbed do meet the performance targets.

An infinitesimally passing testbed resembles a epsilon-delta proof in calculus. Construct a test network such that all of the individual tests of the TIDS pass by only small (infinitesimal) margins, and demonstrate that a variety of authentic applications running over real TCP implementations (or other protocol as appropriate) meets the Target Transport Performance over such a network. The workloads should include multiple types of streaming media and transaction oriented short flows (e.g. synthetic web traffic).

For example, for the HD streaming video TIDS described in Section 9, the IP capacity should be exactly the header overhead above 2.5 Mb/s, the per packet random background loss ratio should be 1/363, for a run length of 363 packets, the bottleneck queue should be 11 packets and the front path should have just enough buffering to withstand 11 packet interface rate bursts. We want every one of the TIDS tests to fail if we slightly increase the relevant test parameter, so for example sending a 12 packet bursts should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. On this infinitesimally passing network it should be possible for a real application using a stock TCP implementation in the vendor's default configuration to attain 2.5 Mb/s over an 50 mS path.

The most difficult part of setting up such a testbed is arranging for it to infinitesimally pass the individual tests. Two approaches: constraining the network devices not to use all available resources (e.g. by limiting available buffer space or data rate); and preloading subpaths with cross traffic. Note that is it important that a single environment be constructed which infinitesimally passes all tests at the same time, otherwise there is a chance that TCP can exploit extra latitude in some parameters (such as data rate) to partially compensate for constraints in other parameters (queue space, or viceversa).

To the extent that a TIDS is used to inform public dialog it should be fully publicly documented, including the details of the tests, what assumptions were used and how it was derived. All of the details of the validation experiment should also be published with sufficient detail for the experiments to be replicated by other researchers. All components should either be open source of fully described proprietary implementations that are available to the research community.

11. Security Considerations

Measurement is often used to inform business and policy decisions, and as a consequence is potentially subject to manipulation. Model Based Metrics are expected to be a huge step forward because

equivalent measurements can be performed from multiple vantage points, such that performance claims can be independently validated by multiple parties.

Much of the acrimony in the Net Neutrality debate is due by the historical lack of any effective vantage independent tools to characterize network performance. Traditional methods for measuring Bulk Transport Capacity are sensitive to RTT and as a consequence often yield very different results when run local to an ISP or internconnect and when run over a customer's complete path. Neither the ISP nor customer can repeat the other's measurements, leading to high levels of distrust and acrimony. Model Based Metrics are expected to greatly improve this situation.

This document only describes a framework for designing Fully Specified Targeted IP Diagnostic Suite. Each FSTIDS MUST include its own security section.

12. Acknowledgements

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length. Alex Gilgur for helping with the statistics.

Meredith Whittaker for improving the clarity of the communications.

Ruediger Geib provided feedback which greatly improved the document.

This work was inspired by Measurement Lab: open tools running on an open platform, using open tools to collect open data. See http://www.measurementlab.net/

13. IANA Considerations

This document has no actions for IANA.

14. References

14.1. Normative References

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

14.2. Informative References

[RFC0863] Postel, J., "Discard Protocol", STD 21, RFC 863, May 1983.

[RFC0864] Postel, J., "Character Generator Protocol", STD 22, RFC 864, May 1983.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[RFC3148] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, February 2003.

[RFC4015] Ludwig, R. and A. Gurtov, "The Eifel Response Algorithm for TCP", RFC 4015, February 2005.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, November 2006.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, May 2007.

[RFC5136] Chimento, P. and J. Ishac, "Defining Network Capacity", RFC 5136, February 2008.

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

[RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, August 2012.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/

RFC6928, April 2013,
<http://www.rfc-editor.org/info/rfc6928>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, August 2014.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, February 2015.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[I-D.ietf-ippm-2680-bis]
Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, "A One-Way Loss Metric for IPPM", draft-ietf-ippm-2680-bis-05 (work in progress), August 2015.

[MSMO97] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communications Review volume 27, number3, July 1997.

[WPING] Mathis, M., "Windowed Ping: An IP Level Performance Diagnostic", INET 94, June 1994.

[mpingSource]
Fan, X., Mathis, M., and D. Hamon, "Git Repository for mping: An IP Level Performance Diagnostic", Sept 2013, <https://github.com/m-lab/mping>.

[MBMSource]
Hamon, D., Stuart, S., and H. Chen, "Git Repository for Model Based Metrics", Sept 2013, <https://github.com/m-lab/MBM>.

[Pathdiag]
Mathis, M., Heffner, J., O'Neil, P., and P. Siemsen, "Pathdiag: Automated TCP Diagnosis", Passive and Active Measurement , June 2008.

[iperf] Wikipedia Contributors, "iPerf", Wikipedia, The Free Encyclopedia , cited March 2015, <http://en.wikipedia.org/w/index.php?title=Iperf&oldid=649720021>.

[StatQC] Montgomery, D., "Introduction to Statistical Quality Control - 2nd ed.", ISBN 0-471-51988-X, 1990.

[Rtool] R Development Core Team, "R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/", , 2011.

[CVST] Krueger, T. and M. Braun, "R package: Fast Cross-Validation via Sequential Testing", version 0.1, 11 2012.

[AFD] Pan, R., Breslau, L., Prabhakar, B., and S. Shenker, "Approximate fairness through differential dropping", SIGCOMM Comput. Commun. Rev. 33, 2, April 2003.

[wikiBloat] Wikipedia, "Bufferbloat", http://en.wikipedia.org/w/index.php?title=Bufferbloat&oldid=608805474, March 2015.

[CCscaling] Fernando, F., Doyle, J., and S. Steven, "Scalable laws for stable network congestion control", Proceedings of Conference on Decision and Control, http://www.ee.ucla.edu/~paganini, December 2001.

[TSO_pacing] Corbet, J., "TSO sizing and the FQ scheduler", LWN.net https://lwn.net/Articles/564978/, Aug 2013.

[TSO_fq_pacing] Dumazet, E. and Y. Chen, "TSO, fair queuing, pacing: three's a charm", Proceedings of IETF 88, TCPM WG https://www.ietf.org/proceedings/88/slides/slides-88-tcpm-9.pdf, Nov 2013.

Appendix A. Model Derivations

The reference target_run_length described in Section 5.2 is based on very conservative assumptions: that all window above target_window_size contributes to a standing queue that raises the RTT, and that classic Reno congestion control with delayed ACKs are in effect. In this section we provide two alternative calculations using different assumptions.

It may seem out of place to allow such latitude in a measurement standard, but this section provides offsetting requirements.

The estimates provided by these models make the most sense if network performance is viewed logarithmically. In the operational Internet, data rates span more than 8 orders of magnitude, RTT spans more than 3 orders of magnitude, and packet loss ratio spans at least 8 orders of magnitude if not more. When viewed logarithmically (as in decibels), these correspond to 80 dB of dynamic range. On an 80 dB scale, a 3 dB error is less than 4% of the scale, even though it represents a factor of 2 in untransformed parameter.

This document gives a lot of latitude for calculating target_run_length, however people designing a TIDS should consider the effect of their choices on the ongoing tussle about the relevance of "TCP friendliness" as an appropriate model for Internet capacity allocation. Choosing a target_run_length that is substantially smaller than the reference target_run_length specified in Section 5.2 strengthens the argument that it may be appropriate to abandon "TCP friendliness" as the Internet fairness model. This gives developers incentive and permission to develop even more aggressive applications and protocols, for example by increasing the number of connections that they open concurrently.

A.1. Queueless Reno

In Section 5.2 models were derived based on the assumption that the subpath IP rate matches the target rate plus overhead, such that the excess window needed for the AIMD sawtooth causes a fluctuating queue at the bottleneck.

An alternate situation would be a bottleneck where there is no significant queue and losses are caused by some mechanism that does not involve extra delay, for example by the use of a virtual queue as done in Approximate Fair Dropping [AFD]. A flow controlled by such a bottleneck would have a constant RTT and a data rate that fluctuates in a sawtooth due to AIMD congestion control. Assume the losses are being controlled to make the average data rate meet some goal which is equal or greater than the target_rate. The necessary run length to meet the target_rate can be computed as follows:

For some value of Wmin, the window will sweep from Wmin packets to 2*Wmin packets in 2*Wmin RTT (due to delayed ACK). Unlike the queueing case where Wmin = target_window_size, we want the average of Wmin and 2*Wmin to be the target_window_size, so the average data rate is the target rate. Thus we want Wmin = (2/3)*target_window_size.

Between losses each sawtooth delivers (1/2)(Wmin+2*Wmin)(2Wmin) packets in 2*Wmin round trip times.

Substituting these together we get:

target_run_length = (4/3)(target_window_size^2)

Note that this is 44% of the reference_run_length computed earlier. This makes sense because under the assumptions in Section 5.2 the AMID sawtooth caused a queue at the bottleneck, which raised the effective RTT by 50%.

Appendix B. The effects of ACK scheduling

For many network technologies simple queueing models don't apply: the network schedules, thins or otherwise alters the timing of ACKs and data, generally to raise the efficiency of the channel allocation algorithms when confronted with relatively widely spaced small ACKs. These efficiency strategies are ubiquitous for half duplex, wireless and broadcast media.

Altering the ACK stream by holding or thinning ACKs typically has two consequences: it raises the implied bottleneck IP capacity, making the fine grained slowstart bursts either faster or larger and it raises the effective RTT by the average time that the ACKs and data are delayed. The first effect can be partially mitigated by reclocking ACKs once they are beyond the bottleneck on the return path to the sender, however this further raises the effective RTT.

The most extreme example of this sort of behavior would be a half duplex channel that is not released as long as the endpoint currently holding the channel has more traffic (data or ACKs) to send. Such environments cause self clocked protocols under full load to revert to extremely inefficient stop and wait behavior. The channel constrains the protocol to send an entire window of data as a single contiguous burst on the forward path, followed by the entire window of ACKs on the return path.

If a particular return path contains a subpath or device that alters the timing of the ACK stream, then the entire front path from the sender up to the bottleneck must be tested at the burst parameters implied by the ACK scheduling algorithm. The most important parameter is the Implied Bottleneck IP Capacity, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACK stream (relying on the cumulative nature of seg.ack to permit discarding some ACKs) requires larger sender interface bursts to offset the longer times between ACK in order to maintain the average data rate.

It is important to note that due to ubiquitous self clocking in

Internet protocols, ill conceived channel allocation mechanisms increases the queueing stress on the front path because they cause larger full sender rate data bursts.

Holding data or ACKs for channel allocation or other reasons (such as forward error correction) always raises the effective RTT relative to the minimum delay for the path. Therefore it may be necessary to replace target_RTT in the calculation in Section 5.2 by an effective_RTT, which includes the target_RTT plus a term to account for the extra delays introduced by these mechanisms.

Appendix C. Version Control

This section to be removed prior to publication.

Formatted: Mon Oct 19 15:59:51 PDT 2015

Authors' Addresses

Matt Mathis
Google, Inc
1600 Amphitheater Parkway
Mountain View, California 94043
USA

Email: mattmathis@google.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ 07748
USA

Phone: +1 732 420 1571
Email: acmorton@att.com
URI: http://home.comcast.net/~acmacm/