IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: September 10, 2015

M. Mathis
Google, Inc
A. Morton
AT&T Labs
March 9, 2015

Model Based Bulk Performance Metrics
draft-ietf-ippm-model-based-metrics-04.txt

Abstract

We introduce a new class of model based metrics designed to determine if an end-to-end Internet path can meet predefined bulk transport performance targets 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, to accurately detect if any part of the infrastructure will prevent the full end-to-end paths traversing them from meeting the specified target performance.

The diagnostic tests consist of precomputed traffic patterns and statistical criteria for evaluating packet delivery. The traffic patterns are precomputed to mimic TCP or other transport protocol over a long path but are constructed in such a way that they are independent of the actual details of the subpath under test, end systems or applications. Likewise the success criteria depends on the packet delivery statistics of the subpath, as evaluated against a protocol model applied to the target performance. The success criteria also does not depend on the details of the subpath, endsystems or application. This makes the measurements open loop, eliminating most of the difficulties encountered by traditional bulk transport metrics.

Model based metrics exhibit several important new properties not present in other Bulk 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 results from multiple Measurement Points.

This document does not define diagnostic tests directly, but provides a framework for designing suites of diagnostics tests that are tailored to confirming that infrastructure can meet the target performance.

Status of this Memo

This Internet-Draft is submitted in full conformance with the

provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Table of Contents

1. Introduction 5
1.1. TODO 7
2. Terminology 7
3. New requirements relative to RFC 2330 11
4. Background 12
4.1. TCP properties 13
4.2. Diagnostic Approach 14
5. Common Models and Parameters 15
5.1. Target End-to-end parameters 16
5.2. Common Model Calculations 16
5.3. Parameter Derating 17
6. Common testing procedures 18
6.1. Traffic generating techniques 18
6.1.1. Paced transmission 18
6.1.2. Constant window pseudo CBR 19
6.1.3. Scanned window pseudo CBR 19
6.1.4. Concurrent or channelized testing 20
6.2. Interpreting the Results 21
6.2.1. Test outcomes 21
6.2.2. Statistical criteria for estimating run_length 22
6.2.3. Reordering Tolerance 24
6.3. Test Preconditions 25
7. Diagnostic Tests 25
7.1. Basic Data Rate and Delivery Statistics Tests 26
7.1.1. Delivery Statistics at Paced Full Data Rate 26
7.1.2. Delivery Statistics at Full Data Windowed Rate 27
7.1.3. Background Delivery Statistics Tests 27
7.2. Standing Queue Tests 27
7.2.1. Congestion Avoidance 29
7.2.2. Bufferbloat 29
7.2.3. Non excessive loss 30
7.2.4. Duplex Self Interference 30
7.3. Slowstart tests 30
7.3.1. Full Window slowstart test 31
7.3.2. Slowstart AQM test 31
7.4. Sender Rate Burst tests 31
7.5. Combined and Implicit Tests 32
7.5.1. Sustained Bursts Test 32
7.5.2. Streaming Media 33
8. An Example 34
9. Validation 36
10. Security Considerations 37
11. Acknowledgements 37
12. IANA Considerations 38
13. References 38
13.1. Normative References 38

13.2. Informative References 38
Appendix A. Model Derivations 40
A.1. Queueless Reno 41
Appendix B. Complex Queueing 42
Appendix C. Version Control 43
Authors' Addresses 43

1. Introduction

Bulk performance metrics evaluate an Internet path's ability to carry bulk data. Model based bulk performance metrics rely on mathematical TCP models to design a targeted diagnostic suite (TDS) of IP performance tests which can be applied independently to each subpath of the full end-to-end path. These targeted diagnostic suites allow independent tests of subpaths to accurately detect if any subpath will prevent the full end-to-end path from delivering bulk data at the specified performance target, independent of the measurement vantage points or other details of the test procedures used for each measurement.

The end-to-end target performance is determined by the needs of the user or application, outside the scope of this document. For bulk data transport, the primary 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 entire end-to-end path over which the data traverses, these parameters must also be specified in advance. They may reflect a specific real path through the Internet or an idealized path representing a typical user community. The target values for these three parameters, Data Rate, RTT and MTU, inform the mathematical models used to design the TDS.

Each IP diagnostic test in a TDS consists of a precomputed traffic pattern and statistical criteria for evaluating packet delivery.

Mathematical models are used to design traffic patterns that mimic TCP or other bulk transport protocol 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 computed in advance based on the three target parameters of the end-to-end path and independent of the properties of individual subpaths. As much as possible the measurement traffic is generated deterministically in ways that minimize the extent to which test methodology, measurement points, measurement vantage or path partitioning affect the details of the measurement traffic.

Mathematical models are also used to compute the bounds on the packet delivery statistics for acceptable IP performance. Since these statistics, such as packet loss, are typically aggregated from all subpaths of the end-to-end path, the end-to-end statistical bounds need to be apportioned as a separate bound for each subpath. Note that links that are expected to be bottlenecks are expected to contribute a larger fraction of the total packet loss and/or delay. In compensation, other links have to be constrained to contribute

less packet loss and delay. The criteria for passing each test of a TDS is an apportioned share of the total bound determined by the mathematical model from the end-to-end target performance.

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.

This document describes a framework for deriving traffic patterns and delivery statistics for model based metrics. It does not fully specify any measurement techniques. Important details such as packet type-p selection, sampling techniques, vantage selection, etc. are not specified here. We imagine Fully Specified Targeted Diagnostic Suites (FSTDS), that define all of these details. We use TDS to refer to the subset of such a specification that is in scope for this document. A TDS includes the target parameters, documentation of the models and assumptions used to derive the diagnostic test parameters, specifications for the traffic and delivery statistics for the tests themselves, and a description of a test setup that can be used to validate the tests and models.

Section 2 defines terminology used throughout this document.

It has been difficult to develop Bulk Transport Capacity [RFC3148] metrics due to some overlooked requirements described in Section 3 and some intrinsic problems with using protocols for measurement, described in Section 4.

In Section 5 we describe the models and common parameters used to derive the targeted diagnostic suite. In Section 6 we describe common testing procedures. Each subpath is evaluated using suite of far simpler and more predictable diagnostic tests described in Section 7. In Section 8 we present an example TDS that might be representative of HD video, and illustrate how MBM 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.

There exists a small risk that model based metric itself might yield a false pass result, in the sense that every subpath of an end-to-end path passes every IP diagnostic test and yet a real application fails to attain the performance target over the end-to-end path. If this happens, then the validation procedure described in Section 9 needs to be used to prove and potentially revise the models.

Future documents may define model based metrics for other traffic

classes and application types, such as real time streaming media.

1.1. TODO

This section to be removed 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 Mar 9 14:37:24 PDT 2015

2. 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].

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

[data] sender: Host sending data and receiving ACKs.
[data] receiver: Host receiving data and sending ACKs.
subpath: A portion of the full path. Note that there is no requirement that subpaths be non-overlapping.
Measurement Point: Measurement points as described in [RFC7398].
test path: A path between two measurement points that includes a subpath of the end-to-end path under test, and could include infrastructure between the measurement points and the subpath.
[Dominant] Bottleneck: The Bottleneck that generally dominates traffic statistics for the entire path. It typically determines a flow's self clock timing, packet loss and ECN marking rate. See Section 4.1.
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 end-to-end path and application. They 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. This is the payload data rate, and excludes transport and lower level headers(TCP/IP or other protocols) and as well as retransmissions and other data that does not contribute to the total quantity of data delivered to the application.

Link Data Rate: General term for the data rate as seen by the link or lower layers. The link data rate includes transport and IP headers, retransmissions and other transport layer overhead. This document is agnostic as to whether the link data rate includes or excludes framing, MAC, or other lower layer overheads, except that they must be treated uniformly.

end-to-end target parameters: Application or transport performance goals for the end-to-end path. They include the target data rate, RTT and MTU described below.

Target Data Rate: The application data rate, typically the ultimate user's performance goal.

Target RTT (Round Trip Time): The baseline (minimum) RTT of the longest end-to-end path over which the application 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 traffic statistics. The Target RTT must be specified considering authentic packets sizes: MTU sized packets on the forward path, ACK sized packets (typically header_overhead) on the return path.

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

Effective Bottleneck Data Rate: This is the bottleneck data rate inferred from the ACK stream, by looking at how much data the ACK stream reports delivered per unit time. If the path is thinning ACKs or batching packets the effective bottleneck rate can be much higher than the average link rate. See Section 4.1 and Appendix B for more details.

[sender | interface] rate: The burst data rate, constrained by the data sender's interfaces. 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. They are

described in more detail in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and link IP performance (includes headers).

pipe size: A general term for number of packets needed in flight (the window size) to exactly fill some network path or subpath. This is the window size which is normally the onset of queueing.

target_pipe_size: The number of packets in flight (the window size) needed to exactly meet the target rate, with a single stream and no cross traffic for the specified application target data rate, RTT, and MTU. It is the amount of circulating data required to meet the target data rate, and 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 (to be) delivered between losses or ECN marks. Nominally one over the loss or ECN marking probability, if there are independently and identically distributed.

target_run_length: The target_run_length is an estimate of the minimum number of good packets needed between losses or ECN 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

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 TDS validation procedures, described in Section 9.

subpath_data_rate: The maximum IP data rate supported by a subpath. This typically includes TCP/IP overhead, including headers, retransmits, etc.

test_path_RTT: The RTT between two measurement points using appropriate data and ACK packet sizes.

test_path_pipe: The amount of data necessary to fill a test path. Nominally the test path RTT times the subpath_data_rate (which should be part of the end-to-end subpath).

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

Tests can be classified into groups according to their applicability.

Capacity tests: determine if a network subpath has sufficient capacity to deliver the target performance. As long as the test traffic is within the proper envelope for the target end-to-end performance, the average packet losses or ECN 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 buffer space), and the test schedules must be balanced by their cost.

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.

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 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 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.

General Terminology:

Targeted Diagnostic Test (TDS): A set of IP Diagnostics designed to determine if a subpath can sustain flows at a specific target_data_rate over a path that has a target_RTT using target_MTU sided packets.

Fully Specified Targeted Diagnostic Test: A TDS 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.

apportioned: To divide and allocate, as in budgeting packet loss rates across multiple subpaths to accumulate below a specified end-to-end loss rate.

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 dependences, such that the resulting dependency graph is acyclic.

Bulk performance metrics: Bulk performance metrics evaluate an Internet path's ability to carry bulk data, such as transporting large files, streaming (non-real time) video, and at some scales, web images and content. (For very fast network, web performance is dominated by pure RTT effects). The metrics presented in this document reflect the evolution of [RFC3148].

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

delivery Statistics: Raw or summary statistics about packet delivery properties of the IP layer including packet losses, ECN marks, reordering, or any other properties that may be germane to transport performance.

IP performance tests: Measurements or diagnostic tests to determine delivery statistics.

3. New requirements relative to RFC 2330

Model Based Metrics are designed to fulfill some additional requirement 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:

- 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.
- Metrics should be spatially composable, such that measures of concatenated paths should be predictable from subpaths. Ideally they should also be differentiable: the metrics of a subpath should be
- 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 portion of the test path that is not under test between the MP and the part that under tests is effectively ideal, or is non ideal in ways that can be calibrated out of the measurements and the test RTT between the MPs is below some reasonable bound.
- Metrics 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 this requirement.

4. Background

At the time the IPPM WG was chartered, sound Bulk Transport Capacity measurement was known to be way beyond our capabilities. By hindsight it is now clear why it 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 network (raise loss probability and/or RTT) to conform to their behavior.
- 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 link may pass a short RTT local test even though it fails when the path is extended by a perfect network to some larger RTT.
- TCP has a meta 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 "elasticity" of the measurement traffic and cross traffic, so you can not even gauge the relative magnitude of their effects on each other.

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

Model Based Metrics overcome these problems by forcing the measurement system to be open loop: the delivery statistics (akin to the network estimators) do not affect the traffic or traffic patterns (bursts), which computed on the basis of the target performance. In order for a network to pass, the resulting delivery statistics and corresponding network estimators have to be such that they would not cause the control systems slow the traffic below the target rate.

4.1. TCP properties

TCP and SCTP are self clocked protocols. The dominant steady state behavior is to have an approximately fixed quantity of data and acknowledgements (ACKs) circulating in the network. The 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 systems is that it is entirely 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 phenomena can cause bursts of data, even in idealized networks that are modeled as simple queueing systems.

During slowstart the data rate is doubled on each RTT by sending twice as much data as was delivered to the receiver on the prior RTT. For slowstart to be able to fill such 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 mark. For example, with classic Reno congestion control, an optimal slowstart has to end with a burst that is twice the bottleneck rate for exactly one RTT in duration. This burst causes a queue which is exactly equal to the pipe size (i.e. the window is exactly twice the pipe size) so when the window is halved in response to the first loss, the new window will be exactly the pipe size.

Note that if the bottleneck data rate is significantly slower than the rest of the path, the slowstart bursts will not cause significant queues anywhere else along the path; they primarily exercise the queue at the dominant bottleneck.

Other sources of bursts include application pauses and channel allocation mechanisms. Appendix B describes the treatment of channel allocation systems. If the application pauses (stops reading or writing data) for some fraction of one RTT, state-of-the-art TCP catches up to the earlier window size by sending a burst of data at the full sender interface rate. To fill such a network with a realistic application, the network has to be able to tolerate interface rate bursts from the data sender large enough to cover application pauses.

Although the interface rate bursts are typically smaller than last burst of a slowstart, they are at a higher data 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 should be tolerated.

To verify that a path can meet a performance target, it is necessary to independently confirm that the path can tolerate bursts in the dimensions that can be caused by these mechanisms. Three cases are likely to be sufficient:

- Slowstart bursts sufficient to get connections started properly.
- Frequent sender interface rate bursts that are small enough where they can be assumed not to significantly affect delivery statistics. (Implicitly derated by selecting the burst size).
- Infrequent sender interface rate full target_pipe_size bursts that do affect the delivery statistics. (Target_run_length may be derated).

4.2. Diagnostic Approach

The MBM approach is to open loop TCP by precomputing traffic patterns that are typically generated by TCP operating at the given target parameters, and evaluating delivery statistics (packet loss, ECN marks and delay). In this approach the measurement software explicitly controls the data rate, transmission pattern or cwnd (TCP's primary congestion control state variables) to create repeatable traffic patterns that mimic TCP behavior but are independent of the actual behavior of the subpath under test. These patterns are manipulated to probe the network to verify that it can deliver all of the traffic patterns that a transport protocol is likely to generate under normal operation at the target rate and RTT.

By opening the protocol control loops, we remove most sources of temporal and spatial correlation in the traffic delivery statistics, such that each subpath's contribution to the end-to-end statistics can be assumed to be independent and stationary (The delivery statistics depend on the fine structure of the data transmissions, but not on long time scale state imbedded in the sender, receiver or other network components.) Therefore each subpath's contribution to the end-to-end delivery statistics can be assumed to be independent, and spatial composition techniques such as [RFC5835] and [RFC6049] apply.

In typical networks, the dominant bottleneck contributes the majority of the packet loss and ECN marks. Often the rest of the path makes insignificant contribution to these properties. A TDS should apportion the end-to-end budget for the specified parameters (primarily packet loss and ECN marks) to each subpath or group of

subpaths. For example the dominant bottleneck may be permitted to contribute 90% of the loss budget, while the rest of the path is only permitted to contribute 10%.

A TDS or FSTDS MUST apportion all relevant packet delivery statistics between successive subpaths, such that the spatial composition of the apportioned metrics will yield end-to-end statics which are within the bounds determined by the models.

A network is expected to be able to sustain a Bulk TCP flow of a given data rate, MTU and RTT when all of the following conditions are met:

1. The raw link rate is higher than the target data rate. See Section 7.1 or any number of data rate tests outside of MBM.
2. The observed packet delivery statistics are better than required by a suitable TCP performance model (e.g. fewer losses or ECN marks). See Section 7.1 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 rate burst large enough to get the flow out of slowstart at a suitable window size. See Section 7.3.
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 7.4.
5. When there is a standing queue at a bottleneck for a shared media subpath (e.g. half duplex), there are suitable bounds on how the data and ACKs interact, for example due to the channel arbitration mechanism. See Section 7.2.4.
6. 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. See Section 7.2.

Note that conditions 1 through 4 require load tests for confirmation, and thus need to be monitored on an ongoing basis. Conditions 5 and 6 require engineering tests. 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].

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 2. These parameters are determined by the needs of the application or the ultimate end user and the end-to-end 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 2 include the effective bottleneck data rate, the sender interface data rate and the TCP/IP header sizes (overhead).

The target data rate must be smaller than all link data rates by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance for fluctuations in the actual data rate, needed to meet the specified average rate. Specifying a target 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 TDS designed for bench testing in the absence of a real application, or for a real physical test, for in situ testing 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 6.1.4 applies.

5.2. Common Model Calculations

The end-to-end target parameters are used to derive the target_pipe_size and the reference target_run_length.

The target_pipe_size, is the average window size in packets needed to meet the target rate, for the specified target RTT and MTU. It is given by:

target_pipe_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 marks, as computed by a mathematical model of TCP congestion control. The derivation here follows [MSMO97], and by design is quite

conservative. The alternate models described in Appendix A generally yield smaller run_lengths (higher acceptable loss or ECN marking rates), but may not apply in all situations. A FSTDS that uses an alternate model MUST compare it to the reference target_run_length computed here.

Reference target_run_length is derived as follows: assume the subpath_data_rate is infinitesimally larger than the target_data_rate plus the required header_overhead. Then target_pipe_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 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_pipe_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_pipe_size)*(2*target_pipe_size) packets, which simplifies to:

target_run_length = 3*(target_pipe_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 TDS or FSTDS 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_pipe_size and target_run_length, directly imply most of the individual parameters for the tests in Section 7.

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 TDS or FSTDS MUST document and justify the actual method used compute the derated metric parameters.
- The validation procedures described in Section 9 must be used to demonstrate the feasibility of meeting the performance targets with infrastructure that infinitesimally passes the derated tests.
- The validation process 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.

6. Common testing procedures

6.1. Traffic generating techniques

6.1.1. Paced transmission

Paced (burst) transmissions: send bursts of data on a timer to meet a particular target rate and pattern. In all cases the specified data rate can either be the application or link rates. Header overheads must be included in the calculations as appropriate.

Headway: Time interval between packets or bursts, 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.

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

Burst: Send sender interface rate bursts on a timer. Specify any 3 of: average rate, packet size, burst size (number of packets) and burst headway (burst start to start). These bursts are typically sent as back-to-back packets at the testers interface rate.

Slowstart bursts: Send 4 packet sender interface rate bursts at an average data rate equal to twice effective bottleneck link rate (but not more than the sender interface rate). This corresponds to the average rate during a TCP slowstart when Appropriate Byte Counting [RFC3465] is present or delayed ack is disabled. Note that if the effective bottleneck link rate is more than half of the sender interface rate, slowstart rate bursts become sender interface rate bursts.

Repeated Slowstart bursts: Slowstart bursts are typically part of larger scale pattern of repeated bursts, such as sending target_pipe_size packets as slowstart bursts on a target_RTT headway (burst start to burst start). Such a stream has three different average rates, depending on the averaging interval. At the finest time scale the average rate is the same as the sender

interface rate, at a medium scale the average rate is twice the effective bottleneck link rate and at the longest time scales the average rate is equal to the target data rate.

Note that in conventional measurement theory, exponential distributions are often used to eliminate many sorts of correlations. For the procedures above, the correlations are created by the network elements and accurately reflect their behavior. At some point in the future, it will be desirable to introduce noise sources into the above pacing models, but they are not warranted at this time.

6.1.2. 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 FSTDS specifying a constant window CBR tests MUST explicitly indicate under what conditions errors in the data cause tests to inconclusive. See the discussion of test outcomes in Section 6.2.1.

Since constant window pseudo CBR testing is sensitive to RTT fluctuations it can not accurately control the data rate in environments with fluctuating delays.

6.1.3. 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 marks. The window is scanned by incrementing it by one packet every 2*target_pipe_size delivered packets. This mimics the additive increase phase of standard TCP congestion avoidance when delayed ACKs are in effect. It normally

separates the the window increases by approximately 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 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 marks. Such a stiffened transport explicitly violates mandatory Internet congestion control and is not suitable for in situ testing. [RFC5681] 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 7.2 describe how to the partition the scans into regions and how to interpret the results.

6.1.4. Concurrent or channelized testing

The procedures described in this document are only directly applicable to single stream performance measurement, e.g. one TCP connection. 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 pin individual flows to specific channels to minimize reordering or other problems and 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 friendliness) 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 uniform response 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. 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 connection might be the only feasible approach.

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

6.2. Interpreting the Results

6.2.1. Test outcomes

To perform an exhaustive test of an end-to-end network path, each test of the TDS is applied to each subpath of an end-to-end path. If any subpath fails any test then an application running over the end-to-end path can also be expected to fail to attain the target 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.

For example consider a test that implements Constant Window Pseudo CBR (Section 6.1.2) by adding rate controls and detailed traffic 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 delivery 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 delivery statistics.

Note that for load tests, if the observed delivery statistics fail to meet the targets, 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 traffic does not depend on network conditions or traffic received. Any mechanism that introduces feedback between the paths measurements and the traffic 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 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 the boundary between passing and inconclusive is at all sensitive to RTT.

One of the goals for evolving TDS 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 TDS or FSTDS.

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 data delivery statistics for deeper study of the behavior of the network path and to measure the tools themselves. Raw delivery statistics can help to drive tool evolution. Under some conditions it might be possible to reevaluate the raw data for satisfying alternate performance targets. However it is important to guard against sampling bias and other implicit feedback which can cause false results and exhibit measurement point vantage sensitivity.

6.2.2. 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 marking probabilities 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 (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 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 mark, or other signal. This constitutes the null Hypothesis:

H0: no more than one mark in target_run_length = 3*(target_pipe_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 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

6.2.3. 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 no 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 history buffer size, MUST be specified in a FSTDS.

6.3. Test Preconditions

Many tests have preconditions which are required to assure their validity. For example the presence or nonpresence of cross traffic on specific subpaths, or 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 about 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 of 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 FSTDS.

7. Diagnostic Tests

The diagnostic tests below are organized by traffic pattern: basic data rate and delivery 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 FSTDS. 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.

7.1. Basic Data Rate and Delivery Statistics Tests

We propose several versions of the basic data rate and delivery statistics test. All measure the number of packets delivered between losses or ECN 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 delivery statistics, is a low rate test designed for ongoing monitoring for changes in subpath quality.

All rely on the receiver accumulating packet delivery statistics as described in Section 6.2.2 to score the outcome:

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

Fail: it is statistically significant that the observed interval between losses or ECN 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 6.3 or neither run length statistical hypothesis was confirmed in the allotted test duration.

7.1.1. 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 6.1.1 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 delivery statistics at full data rate.

7.1.2. 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_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.

Since losses and ECN marks generally cause transport protocols to at least temporarily 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 marks. To pass this test the network MUST deliver target_pipe_size packets in target_RTT time without any losses or ECN marks at least once per two target_pipe_size round trips, in addition to meeting the run length statistical test.

7.1.3. Background Delivery 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 delivery statistics.

7.2. 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 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 losses or ECN marks. For theses technologies, the discussion of queueing does not apply, but it is still required that the onset of losses or ECN marks be at an appropriate point and progressive.

Use the procedure in Section 6.1.3 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_pipe_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 link 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_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.

7.2.1. Congestion Avoidance

A link 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 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 inplace of the onset of queueing. i.e. A link 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 mark.

7.2.2. 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 performance, 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 [RFC2309], [I-D.ietf-aqm-recommendation] and [wikiBloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the link 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 FSTDS.

7.2.3. 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 traffic window increase on the previous RTT. This could be restated as non-decreasing link throughput at the onset of loss, which is easy to meet as long as discarding packets in not more expensive than delivering them. (Note when there is a transient drop in link throughput, outside of a standing queue test, a link that passes other queue tests in this document will have sufficient queue space to hold one RTT worth of data).

Note that conventional Internet traffic 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].

7.2.4. 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 drains 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 forward path 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 some fixed bound above the expected queueing time computed from trom the excess window divided by the link data rate. This bound must be smaller than target_RTT/2 to avoid reverting to stop and wait behavior. (e.g. Packets have to be released at least twice per RTT, to avoid stop and wait behavior.)

7.3. Slowstart tests

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

In general they are 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). Space the bursts such that the average data rate is

equal to the target_data_rate.

7.3.1. 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_pipe_size total packets.

Accumulate packet delivery statistics as described in Section 6.2.2 to score the outcome. Pass if it is statistically significant that the observed number of good packets delivered between losses or ECN marks is larger than the target_run_length. Fail if it is statistically significant that the observed interval between losses or ECN marks is smaller than the target_run_length.

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.

7.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at slowstart_rate), until the first loss, stop, allow the network to drain and repeat, gathering statistics on the last packet delivered before the loss, the loss pattern, 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 and 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 would be best performed on a quiescent network or testbed, since cross traffic has the potential to change the results.

7.4. 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 very 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_pipe_size, so the worst case burst may be 2*target_pipe_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.

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 TDS could include a table of pairs of derating parameters: what burst size to use as a fraction of the target_pipe_size, and how much each burst size is permitted to reduce the run length, relative to to the target_run_length.

7.5. Combined and Implicit Tests

Combined tests efficiently confirm multiple network properties in a single test, possibly as a side effect of normally 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.

7.5.1. Sustained Bursts Test

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

Send target_pipe_size bursts of packets at server interface rate with target_RTT headway (burst start to burst start). Verify that the

observed delivery statistics meets the target_run_length.

Key observations:

- The subpath under test is expected to go idle for some fraction of the time: (subpath_data_rate-target_rate)/subpath_data_rate. 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_pipe_size*derate packet bursts every target_RTT*derate.
- When not derated, this test is the most strenuous load test.
- A link that passes this test is likely to be able to sustain higher rates (close to subpath_data_rate) 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.
- This test by itself is not sufficient: the standing window engineering tests are also needed to ensure that the link is well behaved at and beyond the onset of congestion.
- Assuming the link passes relevant standing window engineering tests (particularly that it has a progressive onset of loss at an appropriate queue depth) the passing sustained burst test is (believed to be) a sufficient verify that the subpath will not impair stream at the target performance under all conditions. Proving this statement will be subject of ongoing research.

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.

7.5.2. Streaming Media

Model Based Metrics can be implicitly implemented as a side effect of serving any non-throughput maximizing traffic, such as streaming media, with some additional controls and instrumentation in the servers. The essential requirement is that the traffic be constrained such that even with arbitrary application pauses, bursts and data rate fluctuations, the traffic stays 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 7.1.2 and Section 7.4 or Section 7.5.1. 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 7.4 and the sender rate burst sizes are implicitly derated by the serving_window_clamp divided by the target_pipe_size at the very least. (Depending on the application behavior, the data traffic might be significantly smoother than specified by any of the burst tests.)

Note that it is important that the target_data_rate be above the actual average rate needed by the application so it can recover after transient pauses caused by congestion or the application itself.

In an alternative implementation the data rate and bursts might be explicitly controlled by a host 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.

8. An Example

In this section a we illustrate a TDS 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_pipe_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 TDS would be to use the sustained burst test, described in Section 7.5.1. 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-ends loss budget, independent subpath tests could be implemented by apportioning the loss rate 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 hight loss rates 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 links 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).

9. 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 TDS can robustly detect links that will prevent authentic applications using state-of-the-art protocol implementations from meeting the specified performance targets. This correctness criteria is potentially difficult to prove, because it implicitly requires validating a TDS against all possible links 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 TDS, 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 infinitessimally passing testbed do meet the performance targets.

An infinitessimally passing testbed resembles a epsilon-delta proof in calculus. Construct a test network such that all of the individual tests of the TDS 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 end-to-end target parameters 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 TDS described in Section 8, the link layer bottleneck data rate should be exactly the header overhead above 2.5 Mb/s, the per packet random background loss probability 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 TDS 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 infinitessimally 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 infinitessimally 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 TDS 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.

10. Security Considerations

Measurement is often used to inform business and policy decisions, and as a consequence is potentially subject to manipulation for illicit gains. 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 local to an ISP and end-to-end. 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 Diagnostic Suite. Each FSTDS MUST include its own security section.

11. 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.

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/

12. IANA Considerations

This document has no actions for IANA.

13. References

13.1. Normative References

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

13.2. 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.

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

[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.

[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.

[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.

[I-D.ietf-aqm-recommendation]
Baker, F. and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management", draft-ietf-aqm-recommendation-11 (work in progress), February 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.

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_pipe_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 loss probability spans at least 8 orders of magnitude. 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 might represent a factor of 2 in untransformed parameter.

This document gives a lot of latitude for calculating target_run_length, however people designing a TDS 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 it was assumed that the link 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 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 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 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_pipe_size, we want the average of Wmin and 2*Wmin to be the target_pipe_size, so the average rate is the target rate. Thus we want Wmin = (2/3)*target_pipe_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_pipe_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. Complex Queueing

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 when confronted with relatively widely spaced small ACKs. These efficiency strategies are ubiquitous for half duplex, wireless and broadcast media.

Altering the ACK stream generally has two consequences: it raises the effective bottleneck data rate, making slowstart burst at higher rates (possibly as high as the sender's interface rate) and it effectively raises the RTT by the average time that the ACKs and data were 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 end point 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, where they send an entire window of data as a single burst of the forward path, followed by the entire window of ACKs on the return path. It is important to note that due to self clocking, ill conceived channel allocation mechanisms can increase the stress on upstream links in a long path: they cause large and faster bursts.

If a particular end-to-end path contains a link or device that alters the ACK stream, then the entire 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 Effective Bottleneck Data Rate, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACKs (relying on the cumulative nature of seg.ack to permit discarding some ACKs) is implies an effectively infinite bottleneck data rate.

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 Mar 9 14:37:24 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/