Internet Engineering Task Force (IETF)
Request for Comments: 8337
Category: Experimental
ISSN: 2070-1721

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

March 2018

Model-Based Metrics for Bulk Transport Capacity

Abstract

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

Model-Based Metrics rely on mathematical models to specify a Targeted IP Diagnostic Suite, a set of IP diagnostic tests designed to assess whether common transport protocols can be expected to meet a predetermined Target Transport Performance over an Internet path.

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

In most cases, the IP diagnostic tests can be implemented by combining existing IPPM metrics with additional controls for generating test streams having a specified temporal structure (bursts or standing queues caused by constant bit rate streams, etc.) and statistical criteria for evaluating packet transfer. The temporal structure of the test streams mimics transport protocol behavior over the complete path; the statistical criteria models the transport protocol's response to less-than-ideal IP packet transfer. In control theory terms, the tests are "open loop". Note that running a test requires the coordinated activity of sending and receiving measurement points.

This document addresses Bulk Transport Capacity. It describes an alternative to the approach presented in "A Framework for Defining Empirical Bulk Transfer Capacity Metrics" [RFC3148]. Other Model-Based Metrics may cover other applications and transports, such as Voice over IP (VoIP) over UDP, RTP, and new transport protocols.

This document assumes a traditional Reno TCP-style, self-clocked, window-controlled transport protocol that uses packet loss and Explicit Congestion Notification (ECN) Congestion Experienced (CE) marks for congestion feedback. There are currently some experimental protocols and congestion control algorithms that are rate based or otherwise fall outside of these assumptions. In the future, these new protocols and algorithms may call for revised models.

The MBM approach, i.e., mapping Target Transport Performance to a Targeted IP Diagnostic Suite (TIDS) of IP tests, solves some intrinsic problems with using TCP or other throughput-maximizing protocols for measurement. In particular, all throughput-maximizing protocols (especially TCP congestion control) cause some level of congestion in order to detect when they have reached the available capacity limitation of the network. This self-inflicted congestion obscures the network properties of interest and introduces non-linear dynamic equilibrium behaviors that make any resulting measurements useless as metrics because they have no predictive value for conditions or paths different from that of the measurement itself. In order to prevent these effects, it is necessary to avoid the effects of TCP congestion control in the measurement method. These issues are discussed at length in Section 4. Readers who are unfamiliar with basic properties of TCP and TCP-like congestion control may find it easier to start at Section 4 or 4.1.

A Targeted IP Diagnostic Suite does not have such difficulties. IP diagnostics can be constructed such that they make strong statistical statements about path properties that are independent of measurement details, such as vantage and choice of measurement points.

2. Overview

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

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

Figure 1 shows the MBM modeling and measurement framework. The Target Transport Performance at the top of the figure is determined by the needs of the user or application, which are outside the scope of this document. For Bulk Transport Capacity, the main performance parameter of interest is the Target Data Rate. However, since TCP's ability to compensate for less-than-ideal network conditions is fundamentally affected by the Round-Trip Time (RTT) and the Maximum Transmission Unit (MTU) of the complete path, these parameters must also be specified in advance based on knowledge about the intended application setting. They may reflect a specific application over a real path through the Internet or an idealized application and hypothetical path representing a typical user community. Section 5 describes the common parameters and models derived from the Target Transport Performance.

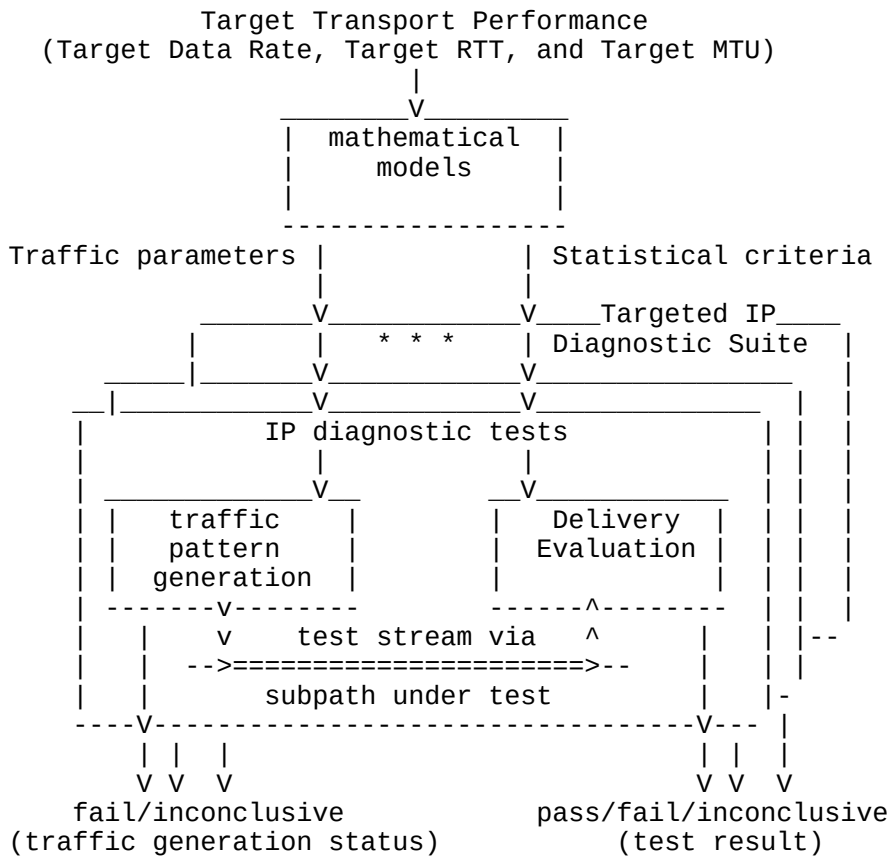

Figure 1: Overall Modeling Framework

Mathematical TCP models are used to determine traffic parameters and subsequently to design traffic patterns that mimic TCP (which has burst characteristics at multiple time scales) or other transport protocols delivering bulk data and operating at the Target Data Rate, MTU, and RTT over a full range of conditions. Using the techniques described in Section 6, the traffic patterns are generated based on the three Target parameters of the complete path (Target Data Rate, Target RTT, and Target MTU), independent of the properties of individual subpaths. As much as possible, the test streams are generated deterministically (precomputed) to minimize the extent to which test methodology, measurement points, measurement vantage, or path partitioning affect the details of the measurement traffic.

Section 7 describes packet transfer statistics and methods to test against the statistical criteria provided by the mathematical models. Since the statistical criteria typically apply to the complete path

(a composition of subpaths) [RFC6049], in situ testing requires that the end-to-end statistical criteria be apportioned as separate criteria for each subpath. Subpaths that are expected to be bottlenecks would then be permitted to contribute a larger fraction of the end-to-end packet loss budget. In compensation, subpaths that are not expected to exhibit bottlenecks must be constrained to contribute less packet loss. Thus, the statistical criteria for each subpath in each test of a TIDS is an apportioned share of the end-to-end statistical criteria for the complete path that was determined by the mathematical model.

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

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

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

3. Terminology

Terms containing underscores (rather than spaces) appear in equations and typically have algorithmic definitions.

3.1. General Terminology

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

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

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

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

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

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

Fully Specified Targeted IP Diagnostic Suite (FSTIDS): A TIDS together with additional specifications such as measurement packet type ("type-p" [RFC2330]) that are out of scope for this document and need to be drawn from other standards documents.

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

determined by the interplay between the network, cross traffic, and congestion control in the transport protocol. It excludes situations where performance is dominated by the RTT alone (e.g., transactions) or bottlenecks elsewhere, such as in the application itself.

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

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

Explicit Congestion Notification (ECN): See [RFC3168].

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

packet loss ratio: As defined in [RFC7680].

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

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

3.2. Terminology about Paths

See [RFC2330] and [RFC7398] for existing terms and definitions.

data sender: Host sending data and receiving ACKs.

data receiver: Host receiving data and sending ACKs.

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

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

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

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

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

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

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

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

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

3.3. Properties

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

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

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

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

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

implied bottleneck IP capacity: The bottleneck IP capacity implied by the ACKs returning from the receiver. It is determined by looking at how much application data the ACK stream at the sender reports as delivered to the data receiver per unit time at various timescales. If the return path is thinning, batching, or otherwise altering the ACK timing, the implied bottleneck IP capacity over short timescales might be substantially larger than the bottleneck IP capacity averaged over a full RTT. Since TCP derives its clock from the data delivered through the bottleneck, the front path must have sufficient buffering to absorb any data bursts at the dimensions (size and IP rate) implied by the ACK stream, which are potentially doubled during slowstart. If the return path is not altering the ACK stream, then the implied bottleneck IP capacity will be the same as the bottleneck IP capacity. See Section 4.1 and Appendix B for more details.

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

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

3.4. Basic Parameters

Basic parameters common to models and subpath tests are defined here. Formulas for target_window_size and target_run_length appear in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and IP performance (includes TCP headers and retransmissions as part of the IP payload).

Network power: The observed data rate divided by the observed RTT. Network power indicates how effectively a transport protocol is filling a network.

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

pipe size: A general term for the number of packets needed in flight (the window size) to exactly fill a network path or subpath. It corresponds to the window size, which maximizes network power. It is often used with additional qualifiers to specify which path, under what conditions, etc.

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

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

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

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

3.5. Ancillary Parameters

The following ancillary parameters are used for some tests:

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

procedures, described in Section 10. Models can be derated by including a multiplicative derating factor to make tests less stringent.

subpath_IP_capacity: The IP capacity of a specific subpath.

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

test_path_RTT: The RTT observed between two measurement points using packet sizes that are consistent with the transport protocol. This is generally MTU-sized packets of the forward path and packets with a size of header_overhead on the return path.

test_path_pipe: The pipe size of a test path. Nominally, it is the test_path_RTT times the test path IP_capacity.

test_window: The smallest window sufficient to meet or exceed the target_rate when operating with a pure self-clock over a test path. The test_window is typically calculated as follows (but see the discussion in Appendix B about the effects of channel scheduling on RTT):

ceiling(target_data_rate * test_path_RTT / (target_MTU - header_overhead))

On some test paths, the test_window may need to be adjusted slightly to compensate for the RTT being inflated by the devices that schedule packets.

3.6. Temporal Patterns for Test Streams

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

packet headway: Time interval between packets, specified from the start of one to the start of the next. For example, if packets are sent with a 1 ms headway, there will be exactly 1000 packets per second.

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

paced single packets: Individual packets sent at the specified rate or packet headway.

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

slowstart rate: Paced bursts of four packets each at an average data rate equal to twice the implied bottleneck IP capacity (but not more than the sender interface rate). This mimics TCP slowstart. This is a two-level burst pattern described in more detail in Section 6.1. If the implied bottleneck IP capacity is more than half of the sender interface rate, the slowstart rate becomes the sender interface rate.

slowstart burst: A specified number of packets in a two-level burst pattern that resembles slowstart. This mimics one round of TCP slowstart.

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

3.7. Tests

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

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

Monitoring tests: Monitoring tests are designed to capture the most important aspects of a capacity test without presenting excessive ongoing load themselves. As such, they may miss some details of

the network's performance but can serve as a useful reduced-cost proxy for a capacity test, for example, to support continuous production network monitoring.

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

4. Background

When "Framework for IP Performance Metrics" [RFC2330] was published in 1998, sound Bulk Transport Capacity (BTC) measurement was known to be well beyond our capabilities. Even when "A Framework for Defining Empirical Bulk Transfer Capacity Metrics" [RFC3148] was published, we knew that we didn't really understand the problem. Now, in hindsight, we understand why assessing BTC is such a difficult problem:

- TCP is a control system with circular dependencies -- everything affects performance, including components that are explicitly not part of the test (for example, the host processing power is not in-scope of path performance tests).

- Congestion control is a dynamic equilibrium process, similar to processes observed in chemistry and other fields. The network and transport protocols find an operating point that balances opposing forces: the transport protocol pushing harder (raising the data rate and/or window) while the network pushes back (raising packet loss ratio, RTT, and/or ECN CE marks). By design, TCP congestion control keeps raising the data rate until the network gives some indication that its capacity has been exceeded by dropping packets or adding ECN CE marks. If a TCP sender accurately fills a path to its IP capacity (e.g., the bottleneck is 100% utilized), then packet losses and ECN CE marks are mostly determined by the TCP sender and how aggressively it seeks additional capacity; they are not determined by the network itself, because the network must send exactly the signals that TCP needs to set its rate.

- TCP's ability to compensate for network impairments (such as loss, delay, and delay variation, outside of those caused by TCP itself) is directly proportional to the number of send-ACK round-trip exchanges per second (i.e., inversely proportional to the RTT). As a consequence, an impaired subpath may pass a short RTT local test even though it fails when the subpath is extended by an effectively perfect network to some larger RTT.

- TCP has an extreme form of the Observer Effect (colloquially known as the "Heisenberg Effect"). 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. In general, for network measurement, you cannot determine even the order of magnitude of the effect. It is possible to construct measurement scenarios where the measurement traffic starves real user traffic, yielding an overly inflated measurement. The inverse is also possible: the user traffic can fill the network, such that the measurement traffic detects only minimal available capacity. In general, you cannot determine which scenario might be in effect, so you cannot gauge the relative magnitude of the uncertainty introduced by interactions with other network traffic.

- As a consequence of the properties listed above, it is difficult, if not impossible, for two independent implementations (hardware or software) of TCP congestion control to produce equivalent performance results [RFC6576] under the same network conditions.

These properties are a consequence of the dynamic equilibrium behavior intrinsic to how all throughput-maximizing protocols interact with the Internet. These protocols rely on control systems based on estimated network metrics to regulate the quantity of data to send into the network. The packet-sending characteristics in turn alter the network properties estimated by the control system metrics, such that there are circular dependencies between every transmission characteristic and every estimated metric. Since some of these dependencies are nonlinear, the entire system is nonlinear, and any change anywhere causes a difficult-to-predict response in network metrics. As a consequence, Bulk Transport Capacity metrics have not fulfilled the analytic framework envisioned in [RFC2330].

Model-Based Metrics overcome these problems by making the measurement system open loop: the packet transfer statistics (akin to the network estimators) do not affect the traffic or traffic patterns (bursts), which are computed on the basis of the Target Transport Performance. A path or subpath meeting the Target Transfer Performance

requirements would exhibit packet transfer statistics and estimated metrics that would not cause the control system to slow the traffic below the Target Data Rate.

4.1. TCP Properties

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

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

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

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

Several common efficiency algorithms also cause bursts. The self-clock is typically applied to groups of packets: the receiver’s delayed ACK algorithm generally sends only one ACK per two data segments. Furthermore, modern senders use TCP segmentation offload

(TSO) to reduce CPU overhead. The sender's software stack builds super-sized TCP segments that the TSO hardware splits into MTU-sized segments on the wire. The net effect of TSO, delayed ACK, and other efficiency algorithms is to send bursts of segments at full sender interface rate.

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

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

Although the sender interface rate bursts are typically smaller than the last burst of a slowstart, they are at a higher IP rate so they potentially exercise queues at arbitrary points along the front path from the data sender up to and including the queue at the dominant bottleneck. It is known that these bursts can hurt network performance, especially in conjunction with other queue pressure; however, we are not aware of any models for estimating the impact or prescribing limits on the size or frequency of sender rate bursts.

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

- Two-level slowstart bursts sufficient to get connections started properly.

- Ubiquitous sender interface rate bursts caused by efficiency algorithms. We assume four packet bursts to be the most common case, since it matches the effects of delayed ACK during slowstart. These bursts should be assumed not to significantly affect packet transfer statistics.

- Infrequent sender interface rate bursts that are the maximum of the full target_window_size and the initial window size (10 segments in [RFC6928]). The target_run_length may be derated for these large fast bursts.

If a subpath can meet the required packet loss ratio for bursts at all of these scales, then it has sufficient buffering at all potential bottlenecks to tolerate any of the bursts that are likely introduced by TCP or other transport protocols.

4.2. Diagnostic Approach

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

1. The IP capacity is above the Target Data Rate by a sufficient margin to cover all TCP/IP overheads. This can be confirmed by the tests described in Section 8.1 or any number of IP capacity tests adapted to implement MBM.

2. The observed packet transfer statistics are better than required by a suitable TCP performance model (e.g., fewer packet losses or ECN CE marks). See Section 8.1 or any number of low- or fixed-rate packet loss tests outside of MBM.

3. There is sufficient buffering at the dominant bottleneck to absorb a slowstart burst large enough to get the flow out of slowstart at a suitable window size. See Section 8.3.

4. There is sufficient buffering in the front path to absorb and smooth sender interface rate bursts at all scales that are likely to be generated by the application, any channel arbitration in the ACK path, or any other mechanisms. See Section 8.4.

5. When there is a slowly rising standing queue at the bottleneck, then the onset of packet loss has to be at an appropriate point (in time or in queue depth) and has to be progressive, for example, by use of Active Queue Management [RFC7567]. See Section 8.2.

6. When there is a standing queue at a bottleneck for a shared media subpath (e.g., a half-duplex link), there must be a suitable bound on the interaction between ACKs and data, for example, due to the channel arbitration mechanism. See Section 8.2.4.

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

A tool that can perform many of the tests is available from [MBMSource].

4.3. New Requirements Relative to RFC 2330

Model-Based Metrics are designed to fulfill some additional requirements that were not recognized at the time RFC 2330 [RFC2330] was published. 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.

- Metrics must be vantage point invariant over a significant range of measurement point choices, including off-path measurement points. The only requirements for Measurement Point (MP) selection should be that the RTT between the MPs is below some reasonable bound and that the effects of the "test leads" connecting MPs to the subpath under test can be calibrated out of the measurements. The latter might be accomplished if the test leads are effectively ideal or their properties can be deducted from the measurements between the MPs. While many tests require that the test leads have at least as much IP capacity as the subpath under test, some do not, for example, the Background Packet Transfer Statistics Tests described in Section 8.1.3.

- Metric measurements should be repeatable by multiple parties with no specialized access to MPs or diagnostic infrastructure. It should be possible for different parties to make the same measurement and observe the same results. In particular, it is important that both a consumer (or the consumer's delegate) and ISP be able to perform the same measurement and get the same result. Note that vantage independence is key to meeting this requirement.

5. Common Models and Parameters

5.1. Target End-to-End Parameters

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

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

The target_data_rate must be smaller than all subpath IP capacities by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance for fluctuations in TCP's actual data rate. Specifying a target_data_rate with insufficient headroom is likely to result in brittle measurements that have little predictive value.

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

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

5.2. Common Model Calculations

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

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

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

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

The reference target_run_length is derived as follows. Assume the subpath_IP_capacity is infinitesimally larger than the target_data_rate plus the required header_overhead. Then, target_window_size also predicts the onset of queuing. A larger window will cause a standing queue at the bottleneck.

Assume the transport protocol is using standard Reno-style Additive Increase Multiplicative Decrease (AIMD) congestion control [RFC5681] (but not Appropriate Byte Counting [RFC3465]) and the receiver is using standard delayed ACKs. Reno increases the window by one packet every pipe size worth of ACKs. With delayed ACKs, this takes two RTTs per increase. To exactly fill the pipe, the spacing of losses must be no closer than when the peak of the AIMD sawtooth reached exactly twice the target_window_size. Otherwise, the multiplicative window reduction triggered by the loss would cause the network to be underfilled. Per [MSMO97] the number of packets between losses must be the area under the AIMD sawtooth. They must be no more frequent than every 1 in ((3/2)*target_window_size)*(2*target_window_size) packets, which simplifies to:

target_run_length = 3*(target_window_size^2)

Note that this calculation is very conservative and is based on a number of assumptions that may not apply. Appendix A discusses these assumptions and provides some alternative models. If a different model is used, an FSTIDS must document the actual method for computing target_run_length and the 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.

Most of the individual parameters for the tests in Section 8 are derived from target_window_size and target_run_length.

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 FSTIDS must document and justify the actual method used to compute the derated metric parameters.

- The validation procedures described in Section 10 must be used to demonstrate the feasibility of meeting the Target Transport Performance with infrastructure that just barely passes the derated tests.

- The validation process for an FSTIDS itself must be documented in such a way that other researchers can duplicate the validation experiments.

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

5.4. Test Preconditions

Many tests have preconditions that are required to assure their validity. Examples include the presence or non-presence of cross traffic on specific subpaths; negotiating ECN; and a test stream preamble of appropriate length to achieve stable access to network resources in the presence of reactive network elements (as defined in Section 1.1 of [RFC7312]). If preconditions are not properly satisfied for some reason, the tests should be considered to be inconclusive. In general, it is useful to preserve diagnostic information as to why the preconditions were not met and any test data that was collected even if it is not useful for the intended test. Such diagnostic information and partial test data may be useful for improving the test or test procedures themselves.

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

Test preconditions and failure actions must be specified in an FSTIDS.

6. Generating Test Streams

Many important properties of Model-Based Metrics, such as vantage independence, are a consequence of using test streams that have temporal structures that mimic TCP or other transport protocols running over a complete path. As described in Section 4.1, self-

clocked protocols naturally have burst structures related to the RTT and pipe size of the complete path. These bursts naturally get larger (contain more packets) as either the Target RTT or Target Data Rate get larger or the Target MTU gets smaller. An implication of these relationships is that test streams generated by running self-clocked protocols over short subpaths may not adequately exercise the queuing at any bottleneck to determine if the subpath can support the full Target Transport Performance over the complete path.

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

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

In conventional measurement practice, stochastic processes are used to eliminate many unintended correlations and sample biases. However, MBM tests are designed to explicitly mimic temporal correlations caused by network or protocol elements themselves. Some portions of these systems, such as traffic arrival (e.g., test scheduling), are naturally stochastic. Other behaviors, such as back-to-back packet transmissions, are dominated by implementation-specific deterministic effects. Although these behaviors always contain non-deterministic elements and might be modeled stochastically, these details typically do not contribute significantly to the overall system behavior. Furthermore, it is known that real protocols are subject to failures caused by network property estimators suffering from bias due to correlation in their own traffic. For example, TCP's RTT estimator used to determine the Retransmit Timeout (RTO), can be fooled by periodic cross traffic or start-stop applications. For these reasons, many details of the test streams are specified deterministically.

It may prove useful to introduce fine-grained noise sources into the models used for generating test streams in an update of Model-Based Metrics, but the complexity is not warranted at the time this document was written.

6.1. Mimicking Slowstart

TCP slowstart has a two-level burst structure as shown in Figure 2. The fine time structure is caused by efficiency algorithms that deliberately batch work (CPU, channel allocation, etc.) to better amortize certain network and host overheads. ACKs passing through the return path typically cause the sender to transmit small bursts

of data at the full sender interface rate. For example, TCP Segmentation Offload (TSO) and Delayed Acknowledgment both contribute to this effect. During slowstart, these bursts are at the same headway as the returning ACKs but are typically twice as large (e.g., have twice as much data) as the ACK reported was delivered to the receiver. Due to variations in delayed ACK and algorithms such as Appropriate Byte Counting [RFC3465], different pairs of senders and receivers produce slightly different burst patterns. Without loss of generality, we assume each ACK causes four packet sender interface rate bursts at an average headway equal to the ACK headway; this corresponds to sending at an average rate equal to twice the effective bottleneck IP rate. Each slowstart burst consists of a series of four packet sender interface rate bursts such that the total number of packets is the current window size (as of the last packet in the burst).

The coarse time structure is due to each RTT being a reflection of the prior RTT. For real transport protocols, each slowstart burst is twice as large (twice the window) as the previous burst but is spread out in time by the network bottleneck, such that each successive RTT exhibits the same effective bottleneck IP rate. The slowstart phase ends on the first lost packet or ECN mark, which is intended to happen after successive slowstart bursts merge in time: the next burst starts before the bottleneck queue is fully drained and the prior burst is complete.

For the diagnostic tests described below, we preserve the fine time structure but manipulate the coarse structure of the slowstart bursts (burst size and headway) to measure the ability of the dominant bottleneck to absorb and smooth slowstart bursts.

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

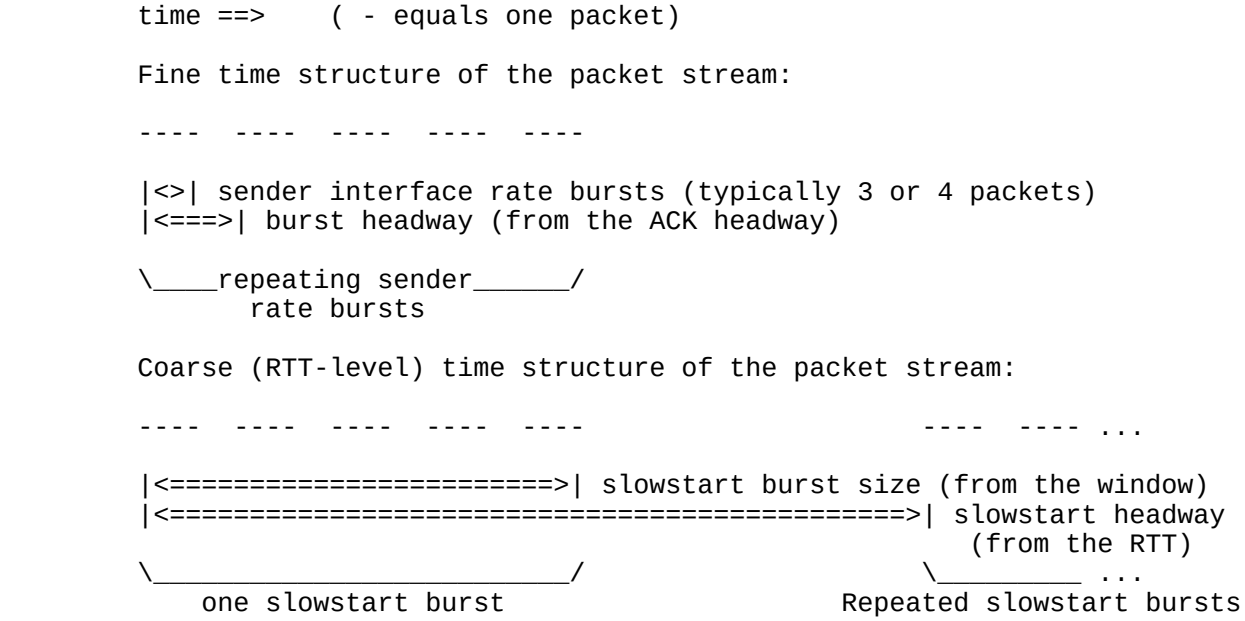

Figure 2: Multiple Levels of Slowstart Bursts

6.2. Constant Window Pseudo CBR

Pseudo constant bit rate (CBR) is implemented by running a standard self-clocked protocol such as TCP with a fixed window size. If that window size is test_window, the data rate will be slightly above the target_rate.

Since the test_window 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 test_window up (as defined above) is likely to 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 (i.e., more demanding for the network path).

Note that Constant Window Pseudo CBR (and Scanned Window Pseudo CBR in the next section) both rely on a self-clock that is at least partially derived from the properties of the subnet under test. This introduces the possibility that the subnet under test exhibits behaviors such as extreme RTT fluctuations that prevent these algorithms from accurately controlling data rates.

An FSTIDS specifying a Constant Window Pseudo CBR test must explicitly indicate under what conditions errors in the data rate cause tests to be inconclusive. Conventional paced measurement traffic may be more appropriate for these environments.

6.3. Scanned Window Pseudo CBR

Scanned Window Pseudo CBR is similar to the Constant Window Pseudo CBR described above, except the window is scanned across a range of sizes designed to include two key events: the onset of queuing and the onset of packet loss or ECN CE marks. The window is scanned by incrementing it by one packet every 2*target_window_size delivered packets. This mimics the additive increase phase of standard Reno TCP congestion avoidance when delayed ACKs are in effect. Normally, the window increases are separated by intervals slightly longer than twice the target_RTT.

There are two ways to implement this test: 1) applying a window clamp to standard congestion control in a standard protocol such as TCP and 2) stiffening a non-standard transport protocol. When standard congestion control is in effect, any losses or ECN CE marks cause the transport to revert to a window smaller than the clamp, such that the scanning clamp loses control of the window size. The NPAD (Network Path and Application Diagnostics) pathdiag tool is an example of this class of algorithms [Pathdiag].

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

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

6.4. Concurrent or Channelized Testing

The procedures described in this document are only directly applicable to single-stream measurement, e.g., one TCP connection or measurement stream. In an ideal world, we would disallow all performance claims based on multiple concurrent streams, but this is not practical due to at least two issues. First, many very high-rate link technologies are channelized and at last partially pin the flow-to-channel mapping to minimize packet reordering within flows.

Second, TCP itself has scaling limits. Although the former problem might be overcome through different design decisions, the latter problem is more deeply rooted.

All congestion control algorithms that are philosophically aligned with [RFC5681] (e.g., claim some level of TCP compatibility, friendliness, or fairness) have scaling limits; that is, as a long fat network (LFN) with a fixed RTT and MTU gets faster, these congestion control algorithms get less accurate and, as a consequence, have difficulty filling the network [CCscaling]. These properties are a consequence of the original Reno AIMD congestion control design and the requirement in [RFC5681] that all transport protocols have similar responses to congestion.

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

If multiple flows are deemed necessary to meet aggregate performance targets, then this must be stated both in the design of the TIDS and in any claims about network performance. The IP diagnostic tests must be performed concurrently with the specified number of connections. For the tests that use bursty test streams, the bursts should be synchronized across streams unless there is a priori knowledge that the applications have some explicit mechanism to stagger their own bursts. In the absence of an explicit mechanism to stagger bursts, many network and application artifacts will sometimes implicitly synchronize bursts. A test that does not control burst synchronization may be prone to false pass results for some applications.

7. Interpreting the Results

7.1. Test Outcomes

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

In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons. Proper instrumentation and treatment of inconclusive outcomes is critical to the accuracy and

robustness of Model-Based Metrics. Tests can be inconclusive if the precomputed traffic pattern or data rates were not accurately generated; the measurement results were not statistically significant; the required preconditions for the test were not met; or other causes. See Section 5.4.

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

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

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

Inconclusive tests may be caused by situations in which a test outcome is ambiguous because of network limitations or an unknown limitation on the IP diagnostic test itself, which may have been caused by some uncontrolled feedback from the network.

Note that procedures that attempt to search the target parameter space to find the limits on a parameter such as target_data_rate are at risk of breaking the location-independent properties of Model-Based Metrics if any part of the boundary between passing, inconclusive, or failing results is sensitive to RTT (which is normally the case). For example, the maximum data rate for a marginal link (e.g., exhibiting excess errors) is likely to be sensitive to the test_path_RTT. The maximum observed data rate over the test path has very little value for predicting the maximum rate over a different path.

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

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

It may be useful to keep raw packet transfer statistics and ancillary metrics [RFC3148] for deeper study of the behavior of the network path and to measure the tools themselves. Raw packet transfer statistics can help to drive tool evolution. Under some conditions, it might be possible to re-evaluate the raw data for satisfying alternate Target Transport Performance. However, it is important to guard against sampling bias and other implicit feedback that can cause false results and exhibit measurement point vantage sensitivity. Simply applying different delivery criteria based on a different Target Transport Performance is insufficient if the test traffic patterns (bursts, etc.) do not match the alternate Target Transport Performance.

7.2. Statistical Criteria for Estimating run_length

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

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

As each packet is sent and measured, we have an ongoing estimate of the performance in terms of the ratio of packet loss or ECN CE marks 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, one mark per target_run_length, where a "mark" is defined as a lost packet, a packet with ECN CE mark, or other signal. This constitutes the null hypothesis:

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

We can stop sending packets if ongoing measurements support accepting H0 with the specified Type I error = alpha (= 0.05, for example).

We also have an alternative hypothesis to evaluate: is performance 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 this document; 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) [Wald45] [Montgomery90]. Note that as originally framed, the events under consideration were all manufacturing defects. In networking, ECN CE marks and lost packets are not defects but signals, indicating that the transport protocol should slow down.

The Sequential Probability Ratio Test also starts with a pair of hypotheses 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) < Xr: continue testing
- defect_count(n) <= Xa: Accept H0
- defect_count(n) >= Xr: 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

Note that the derivations in [Wald45] and [Montgomery90] differ. Montgomery's simplified derivation of SPRT may assume a Bernoulli processes, where the packet loss probabilities are independent and identically distributed, making the SPRT more accessible. Wald's seminal paper showed that this assumption is not necessary. It helps to remember that the goal of SPRT is not to estimate the value of the packet loss rate but only whether or not the packet loss ratio is likely (1) low enough (when we accept the H0 null hypothesis), yielding success or (2) too high (when we accept the H1 alternate hypothesis), yielding failure.

7.3. Reordering Tolerance

All tests must be instrumented for packet-level reordering [RFC4737]. However, there is no consensus for how much reordering should be acceptable. Over the last two decades, the general trend has been to

make protocols and applications more tolerant to reordering (for example, see [RFC5827]), in response to the gradual increase in reordering in the network. This increase has been due to the deployment of technologies such as multithreaded routing lookups and Equal-Cost Multipath (ECMP) routing. These techniques increase parallelism in the network and are critical to enabling overall Internet growth to exceed Moore's Law.

With transport retransmission strategies, there are fundamental trade-offs among reordering tolerance, how quickly losses can be repaired, and overhead from spurious retransmissions. In advance of new retransmission strategies, we propose the following strawman: transport protocols should be able to adapt to reordering as long as the reordering extent is not more than the maximum of one quarter window or 1 ms, whichever is larger. (These values come from experience prototyping Early Retransmit [RFC5827] and related algorithms. They agree with the values being proposed for "RACK: a time-based fast loss detection algorithm" [RACK].) Within this limit on reorder extent, there should be no bound on reordering density.

By implication, recording that 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 (because of the bound on history buffers) should be recorded with the measurement results.

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

8. IP Diagnostic Tests

The IP diagnostic tests below are organized according to the technique used to generate the test stream as described in Section 6. All of the results are evaluated in accordance with Section 7, possibly with additional test-specific criteria.

We also introduce some combined tests that are more efficient when networks are expected to pass but conflate diagnostic signatures when they fail.

8.1. Basic Data Rate and Packet Transfer Tests

We propose several versions of the basic data rate and packet transfer statistics test that differ in how the data rate is controlled. The data can be paced on a timer or window controlled (and self-clocked). The first two tests implicitly confirm that

sub_path has sufficient raw capacity to carry the target_data_rate. They are recommended for relatively infrequent testing, such as an installation or periodic auditing process. The third test, Background Packet Transfer Statistics, is a low-rate test designed for ongoing monitoring for changes in subpath quality.

8.1.1. Delivery Statistics at Paced Full Data Rate

This test confirms 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 Section 6.1 with a burst size of 1 (single packets) or 2 (packet pairs).

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

RFC 6673 [RFC6673] is appropriate for measuring packet transfer statistics at full data rate.

8.1.2. Delivery Statistics at Full Data Windowed Rate

This test confirms 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 test_window.

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

8.1.3. Background Packet Transfer Statistics Tests

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

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

8.2. Standing Queue Tests

These engineering tests confirm that the bottleneck is well behaved across the onset of packet loss, which typically follows after the onset of queuing. 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 or ECN CE marks to signal to the transport protocol that it should reduce its window or data rate. 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 not have an appropriate AQM [RFC7567] and, as a consequence, be subject to bufferbloat [wikiBloat]. Queues without AQM have the potential to inflict excess delays on all flows sharing the bottleneck. Excess losses (more than half of the window) at the onset of loss make loss recovery problematic for the transport protocol. Non-linear, erratic, or excessive RTT increases suggest poor interactions between the channel acquisition algorithms and the transport self-clock. All of the tests in this section use the same basic scanning algorithm, described here, but score the link or subpath on the basis of how well it avoids each of these problems.

Some network technologies rely on virtual queues or other techniques to meter traffic without adding any queuing delay, in which case the data rate will vary with the window size all the way up to the onset of load-induced packet loss or ECN CE marks. For these technologies, the discussion of queuing in Section 6.3 does not apply, but it is still necessary to confirm that the onset of losses or ECN CE marks be at an appropriate point and progressive. If the network bottleneck does not introduce significant queuing delay, modify the procedure described in Section 6.3 to start the scan at a window equal to or slightly smaller than the test_window.

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

Below the onset of queuing, the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the subpath IP rate, the data rate becomes fairly constant, and the RTT increases in proportion to the increase in window size. The precise transition across the start of queuing 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 is taken as the start of the queuing region.

If there is random background loss (e.g., bit errors), precise determination of the onset of queue-induced packet loss may require multiple scans. At window sizes large enough to cause loss in queues, 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 case of stiffened transport protocols (with non-standard, aggressive congestion control algorithms), the details of periodic losses will be dominated by how the window increase function responds to loss.

8.2.1. Congestion Avoidance

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

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

8.2.2. Bufferbloat

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

The test will pass if the onset of loss occurs before a standing queue has introduced delay greater than twice the target_RTT or another 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 queuing delay that is between one and two times the target_RTT.

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

8.2.3. Non-excessive Loss

This test confirms that the onset of loss is not excessive. The test will pass if losses are equal to or less than the increase in the cross traffic plus the test stream window increase since the previous RTT. This could be restated as non-decreasing total throughput of the subpath at the onset of loss. (Note that when there is a transient drop in subpath throughput and there is not already a standing queue, a subpath that passes other queue tests in this document will have sufficient queue space to hold one full RTT worth of data).

Note that token bucket policers will not pass this test, which is as intended. TCP often stumbles badly if more than a small fraction of the packets are dropped in one RTT. Many TCP implementations will require a timeout and slowstart to recover their self-clock. Even if they can recover from the massive losses, the sudden change in available capacity at the bottleneck wastes serving and front-path capacity until TCP can adapt to the new rate [Policing].

8.2.4. Duplex Self-Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path when they share a half-duplex link.

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

The Duplex Self-Interference Test fails if the RTT rises by more than a fixed bound above the expected queuing time computed from the excess window divided by the subpath IP capacity. This bound must be smaller than target_RTT/2 to avoid reverting to stop-and-wait behavior (e.g., data packets and ACKs both have to be released at least twice per RTT).

8.3. Slowstart Tests

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

8.3.1. Full Window Slowstart Test

This capacity test confirms that slowstart is not likely to exit prematurely. To perform this test, send slowstart bursts that are target_window_size total packets and accumulate packet transfer statistics as described in Section 7.2 to score the outcome. The test will pass if it is statistically significant that the observed number of good packets delivered between losses or ECN CE marks is larger than the target_run_length. The test will fail if it is statistically significant that the observed interval between losses or ECN CE marks is smaller than the target_run_length.

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

Note that these are the same parameters that are used for the Sustained Full-Rate Bursts Test, except the burst rate is at slowstart rate rather than sender interface rate.

8.3.2. Slowstart AQM Test

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

This engineering test should be performed on a quiescent network or testbed, since cross traffic has the potential to change the results in ill-defined ways.

8.4. Sender Rate Burst Tests

These tests determine how well the network can deliver bursts sent at the sender's interface rate. Note that this test most heavily exercises the front path and is likely to include infrastructure that 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 about sender interface rate bursts that are not fully defined here. These details, such as the assumed sender interface rate, should be explicitly stated in an FSTIDS.

Current standards permit TCP to send full window bursts following an application pause. (Congestion Window Validation [RFC2861] and updates to support Rate-Limited Traffic [RFC7661] are not required). Since full window bursts are consistent with standard behavior, it is desirable that the network be able to deliver such bursts; otherwise, application pauses will cause unwarranted losses. Note that the AIMD sawtooth requires a peak window that is twice target_window_size, so the worst-case burst may be 2*target_window_size.

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

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

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

8.5. Combined and Implicit Tests

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

8.5.1. Sustained Full-Rate Bursts Test

The Sustained Full-Rate Bursts Test implements a combined worst-case version of all of the capacity tests above. To perform this test, send target_window_size bursts of packets at server interface rate with target_RTT burst headway (burst start to next burst start), and verify that the observed packet transfer statistics meets the target_run_length.

Key observations:

- The subpath under test is expected to go idle for some fraction of the time, determined by the difference between the time to drain the queue at the subpath_IP_capacity and the target_RTT. If the queue does not drain completely, it may be an indication that the subpath has insufficient IP capacity or that there is some other problem with the test (e.g., it is inconclusive).

- The burst sensitivity can be derated by sending smaller bursts more frequently (e.g., by sending target_window_size*derate packet bursts every target_RTT*derate, where "derate" is less than one).

- When not derated, this test is the most strenuous capacity test.

- A subpath that passes this test is likely to be able to sustain higher rates (close to subpath_IP_capacity) for paths with RTTs significantly smaller than the target_RTT.

- This test can be implemented with instrumented TCP [RFC4898], using a specialized measurement application at one end (e.g., [MBMSource]) and a minimal service at the other end (e.g., [RFC863] and [RFC864]).

- This test is efficient to implement, since it does not require per-packet timers, and can make use of TSO in modern network interfaces.

- o If a subpath is known to pass the standing queue engineering tests (particularly that it has a progressive onset of loss at an appropriate queue depth), then the Sustained Full-Rate Bursts Test is sufficient to assure that the subpath under test will not impair Bulk Transport Capacity at the target performance under all conditions. See Section 8.2 for a discussion of the standing queue tests.

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

8.5.2. Passive Measurements

Any non-throughput-maximizing application, such as fixed-rate streaming media, can be used to implement passive or hybrid (defined in [RFC7799]) versions of Model-Based Metrics with some additional instrumentation and possibly a traffic shaper or other controls in the servers. The essential requirement is that the data transmission be constrained such that even with arbitrary application pauses and bursts, the data rate and burst sizes stay within the envelope defined by the individual tests described above.

If the application's serving data rate can be constrained to be 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 serving_window_clamp (which is set to the test_window and computed for the actual serving path).

Under the above constraints, the serving_window_clamp will limit both the serving data rate and burst sizes to be no larger than the parameters specified by the procedures in Section 8.1.2, 8.4, or 8.5.1. Since the serving RTT is smaller than the target_RTT, the worst-case bursts that might be generated under these conditions will be smaller than called for by Section 8.4, and the sender rate burst sizes are implicitly derated by the serving_window_clamp divided by the target_window_size at the very least. (Depending on the application behavior, the data might be significantly smoother than specified by any of the burst tests.)

In an alternative implementation, the data rate and bursts might be explicitly controlled by a programmable traffic shaper or by pacing at the sender. This would provide better control over transmissions but is more complicated to implement, although the required technology is available [TSO_pacing] [TSO_fq_pacing].

Note that these techniques can be applied to any content delivery that can be operated at a constrained data rate to inhibit TCP equilibrium behavior.

Furthermore, note that Dynamic Adaptive Streaming over HTTP (DASH) is generally in conflict with passive Model-Based Metrics measurement, because it is a rate-maximizing protocol. It can still meet the requirement here if the rate can be capped, for example, by knowing a priori the maximum rate needed to deliver a particular piece of content.

9. Example

In this section, we illustrate a TIDS designed to confirm that an access ISP can reliably deliver HD video from multiple content providers to all of its customers. With modern codecs, minimal HD video (720p) generally fits in 2.5 Mb/s. Due to the ISP's geographical size, network topology, and modem characteristics, the ISP determines that most content is within a 50 ms RTT of its users. (This example RTT is sufficient to cover the propagation delay to continental Europe or to either coast of the United States with low-delay modems; it is sufficient to cover somewhat smaller geographical regions if the modems require additional delay to implement advanced compression and error recovery.)

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

Table 1: 2.5 Mb/s over a 50 ms Path

Table 1 shows the default TCP model with no derating and, as such, is quite conservative. The simplest TIDS would be to use the Sustained Full-Rate Bursts Test, described in Section 8.5.1. Such a test would send 11 packet bursts every 50 ms and confirm that there was no more than 1 packet loss per 33 bursts (363 total packets in 1.650 seconds).

Since this number represents the entire end-to-end loss budget, independent subpath tests could be implemented by apportioning the packet loss ratio across subpaths. For example, 50% of the losses might be allocated to the access or last mile link to the user, 40% to the network interconnections 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.

9.1. Observations about Applicability

Guidance on deploying and using MBM belong in a future document. However, the example above illustrates some of the issues that may need to be considered.

Note that another ISP, with different geographical coverage, topology, or modem technology may need to assume a different target_RTT and, as a consequence, a different target_window_size and target_run_length, even for the same target_data rate. One of the implications of this is that infrastructure shared by multiple ISPs, such as Internet Exchange Points (IXPs) and other interconnects may need to be evaluated on the basis of the most stringent target_window_size and target_run_length of any participating ISP. One way to do this might be to choose target parameters for evaluating such shared infrastructure on the basis of a hypothetical reference path that does not necessarily match any actual paths.

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 due to the small RTT (often less than 1 ms). With a short RTT, these tools are likely to report inflated data rates because on a short RTT, these tools can tolerate very high packet loss ratios and can push other cross traffic off of the network. As a consequence, these measurements are useless for predicting actual user performance over longer paths and may themselves be quite disruptive. Model-Based Metrics solves this problem. The interconnect can be evaluated with the same TIDS as other subpaths. Continuing our example, if the interconnect is apportioned 40% of the losses, 11 packet bursts sent every 50 ms should have fewer than one loss per 82 bursts (902 packets).

10. Validation

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

We suggest two approaches, both of which should be applied. First, publish a fully open description of the TIDS, including what assumptions were used and how it was derived, such that the research community can evaluate the design decisions, test them, and comment on their applicability. Second, demonstrate that applications do meet the Target Transport Performance when running over a network testbed that has the tightest possible constraints that still allow the tests in the TIDS to pass.

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

For example, for the HD streaming video TIDS described in Section 9, the IP capacity should be exactly the header_overhead above 2.5 Mb/s, the per packet random background loss ratio should be 1/363 (for a run length of 363 packets), the bottleneck queue should be 11 packets, and the front path should have just enough buffering to withstand 11 packet interface rate bursts. We want every one of the TIDS tests to fail if we slightly increase the relevant test parameter, so, for example, sending a 12-packet burst should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. This network has the tightest possible constraints that can be expected to pass the TIDS, yet 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 a 50 ms path.

The most difficult part of setting up such a testbed is arranging for it to have the tightest possible constraints that still allow it to pass the individual tests. Two approaches are suggested:

- constraining (configuring) the network devices not to use all available resources (e.g., by limiting available buffer space or data rate)

- pre-loading subpaths with cross traffic

Note that it is important that a single tightly constrained environment just barely passes all tests; 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 (e.g., queue space). This effect is potentially bidirectional: extra latitude in the queue space tests has the potential to enable TCP to compensate for insufficient data-rate headroom.

To the extent that a TIDS is used to inform public dialog, it should be fully documented publicly, 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 be either open source or fully described proprietary implementations that are available to the research community.

11. Security Considerations

Measurement is often used to inform business and policy decisions and, as a consequence, is potentially subject to manipulation. Model-Based Metrics are expected to be a huge step forward because equivalent measurements can be performed from multiple vantage points, such that performance claims can be independently validated by multiple parties.

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

Note that in situ measurements sometimes require sending synthetic measurement traffic between arbitrary locations in the network and, as such, are potentially attractive platforms for launching DDoS

attacks. All active measurement tools and protocols must be designed to minimize the opportunities for these misuses. See the discussion in Section 7 of [RFC7594].

Some of the tests described in this document are not intended for frequent network monitoring since they have the potential to cause high network loads and might adversely affect other traffic.

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

12. IANA Considerations

This document has no IANA actions.

13. Informative References

[RFC863] Postel, J., "Discard Protocol", STD 21, RFC 863, DOI 10.17487/RFC0863, May 1983, <https://www.rfc-editor.org/info/rfc863>.

[RFC864] Postel, J., "Character Generator Protocol", STD 22, RFC 864, DOI 10.17487/RFC0864, May 1983, <https://www.rfc-editor.org/info/rfc864>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, DOI 10.17487/RFC2861, June 2000, <https://www.rfc-editor.org/info/rfc2861>.

[RFC3148] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, DOI 10.17487/RFC3148, July 2001, <https://www.rfc-editor.org/info/rfc3148>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, DOI 10.17487/RFC3465, February 2003, <https://www.rfc-editor.org/info/rfc3465>.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, DOI 10.17487/RFC4737, November 2006, <https://www.rfc-editor.org/info/rfc4737>.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, DOI 10.17487/RFC4898, May 2007, <https://www.rfc-editor.org/info/rfc4898>.

[RFC5136] Chimento, P. and J. Ishac, "Defining Network Capacity", RFC 5136, DOI 10.17487/RFC5136, February 2008, <https://www.rfc-editor.org/info/rfc5136>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC5827] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)", RFC 5827, DOI 10.17487/RFC5827, May 2010, <https://www.rfc-editor.org/info/rfc5827>.

[RFC5835] Morton, A., Ed. and S. Van den Berghe, Ed., "Framework for Metric Composition", RFC 5835, DOI 10.17487/RFC5835, April 2010, <https://www.rfc-editor.org/info/rfc5835>.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, DOI 10.17487/RFC6049, January 2011, <https://www.rfc-editor.org/info/rfc6049>.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <https://www.rfc-editor.org/info/rfc6576>.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, DOI 10.17487/RFC6673, August 2012, <https://www.rfc-editor.org/info/rfc6673>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP’s Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/info/rfc6928>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, DOI 10.17487/RFC7312, August 2014, <https://www.rfc-editor.org/info/rfc7312>.

[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, DOI 10.17487/RFC7398, February 2015, <https://www.rfc-editor.org/info/rfc7398>.

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

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <https://www.rfc-editor.org/info/rfc7594>.

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <https://www.rfc-editor.org/info/rfc7661>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <https://www.rfc-editor.org/info/rfc7680>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[AFD] Pan, R., Breslau, L., Prabhakar, B., and S. Shenker, "Approximate fairness through differential dropping", ACM SIGCOMM Computer Communication Review, Volume 33, Issue 2, DOI 10.1145/956981.956985, April 2003.

[CCscaling] Paganini, F., Doyle, J., and S. Low, "Scalable laws for stable network congestion control", Proceedings of IEEE Conference on Decision and Control,, DOI 10.1109/CDC.2001.980095, December 2001.

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

[iPerf] Wikipedia, "iPerf", November 2017, <https://en.wikipedia.org/w/index.php?title=Iperf&oldid=810583885>.

[MBMSource] "mbm", July 2016, <https://github.com/m-lab/MBM>.

[Montgomery90] Montgomery, D., "Introduction to Statistical Quality Control", 2nd Edition, ISBN 0-471-51988-X, 1990.

[mpingSource] "mping", July 2016, <https://github.com/m-lab/mping>.

[MSMO97] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communications Review, Volume 27, Issue 3, DOI 10.1145/263932.264023, July 1997.

[Pathdiag] Mathis, M., Heffner, J., O'Neil, P., and P. Siemsen, "Pathdiag: Automated TCP Diagnosis", Passive and Active Network Measurement, Lecture Notes in Computer Science, Volume 4979, DOI 10.1007/978-3-540-79232-1_16, 2008.

[Policing] Flach, T., Papageorge, P., Terzis, A., Pedrosa, L., Cheng, Y., Karim, T., Katz-Bassett, E., and R. Govindan, "An Internet-Wide Analysis of Traffic Policing", Proceedings of ACM SIGCOMM, DOI 10.1145/2934872.2934873, August 2016.

[RACK] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", Work in Progress, draft-ietf-tcpm-rack-03, March 2018.

[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, 2011, <http://www.R-project.org/>.

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

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

[Wald45] Wald, A., "Sequential Tests of Statistical Hypotheses", The Annals of Mathematical Statistics, Volume 16, Number 2, pp. 117-186, June 1945, <http://www.jstor.org/stable/2235829>.

[wikiBloat]
Wikipedia, "Bufferbloat", January 2018, <https://en.wikipedia.org/w/index.php?title=Bufferbloat&oldid=819293377>.

[WPING] Mathis, M., "Windowed Ping: An IP Level Performance Diagnostic", Computer Networks and ISDN Systems, Volume 27, Issue 3, DOI 10.1016/0169-7552(94)90119-8, June 1994.

Appendix A. Model Derivations

The reference target_run_length described in Section 5.2 is based on very conservative assumptions: that all excess data in flight (i.e., the window size) above the target_window_size contributes to a standing queue that raises the RTT and that classic Reno congestion control with delayed ACKs is 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 method, 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 eight orders of magnitude, RTT spans more than three orders of magnitude, and packet loss ratio spans at least eight orders of magnitude if not more. When viewed logarithmically (as in decibels), these correspond to 80 dB of dynamic range. On an 80 dB scale, a 3 dB error is less than 4% of the scale, even though it represents a factor of 2 in untransformed parameter.

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

A.1. Queueless Reno

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

An alternate situation would be a bottleneck where there is no significant queue and losses are caused by some mechanism that does not involve extra delay, for example, by the use of a virtual queue as done in Approximate Fair Dropping [AFD]. A flow controlled by such a bottleneck would have a constant RTT and a data rate that fluctuates in a sawtooth due to AIMD congestion control. Assume the

losses are being controlled to make the average data rate meet some goal that is equal to or greater than the target_rate. The necessary run length to meet the target_rate can be computed as follows:

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

Between losses, each sawtooth delivers (1/2)(Wmin+2*Wmin)(2Wmin) packets in 2*Wmin RTTs.

Substituting these together, we get:

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

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

Appendix B. The Effects of ACK Scheduling

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

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

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

contiguous burst on the forward path, followed by the entire window of ACKs on the return path. (A channel with this behavior would fail the Duplex Self-Interference Test described in Section 8.2.4).

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

Note that due to ubiquitous self-clocking in Internet protocols, ill-conceived channel allocation mechanisms are likely to increases the queuing stress on the front path because they cause larger full sender rate data bursts.

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

Acknowledgments

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length. Alex Gilgur and Merry Mou helped with the statistics.

Meredith Whittaker improved the clarity of the communications.

Ruediger Geib provided feedback that greatly improved the document.

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

Authors' Addresses

Matt Mathis
Google, Inc
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America

Email: mattmathis@google.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ 07748
United States of America

Phone: +1 732 420 1571
Email: acmorton@att.com