IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: April 24, 2014

M. Mathis
Google, Inc
A. Morton
AT&T Labs
October 21, 2013

Model Based Bulk Performance Metrics
draft-ietf-ippm-model-based-metrics-01.txt

Abstract

We introduce a new class of model based metrics designed to determine if a long network path can meet predefined end-to-end application performance targets by applying a suite of IP diagnostic tests to successive subpaths. The subpath at a time tests are designed to exclude all known conditions which might prevent the full end-to-end path from meeting the user's target application performance.

This approach makes it possible to to determine the IP performance requirements needed to support the desired end-to-end TCP performance. The IP metrics are based on traffic patterns that mimic TCP or other transport protocol but are precomputed independently of the actual behavior of the transport protocol over the subpath under test. This makes the measurements open loop, eliminating nearly all of the difficulties encountered by traditional bulk transport metrics, which fundamentally depend on congestion control equilibrium behavior.

A natural consequence of this methodology is verifiable network measurement: measurements from any given vantage point can be verified by repeating them from other vantage points.

Formatted: Mon Oct 21 15:42:35 PDT 2013

material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2014.

Copyright Notice

Table of Contents

1. Introduction

Model based bulk performance metrics evaluate an Internet path's ability to carry bulk data. TCP models are used to design a targeted diagnostic suite (TDS) of IP performance tests which can be applied independently to each subpath of the full end-to-end path. A targeted diagnostic suite is constructed such that independent tests of the subpaths will accurately predict if the full end-to-end path can deliver bulk data at the specified performance target, independent of the measurement vantage points or other details of the test procedures used to measure each subpath.

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

TCP models are used to design traffic patterns that mimic TCP or other bulk transport protocol operating at the target performance and RTT over a full range of conditions, including flows that are bursty at multiple time scales. The traffic patterns are computed in advance based on the properties of the full end-to-end path and independent of the properties of individual subpaths. As much as possible the traffic is generated deterministically in ways that minimizes the extent to which test methodology, measurement points, measurement vantage or path partitioning effect the details of the traffic.

Models are also used to compute the bounds on the packet delivery statistics for acceptable the IP performance. The criteria for passing each test are determined from the end-to-end target performance and are independent of the subpath under test. In addition to passing or failing, a test can be inconclusive if the precomputed traffic pattern was not authentically generated, test preconditions were not met or the measurement results were not statistically significant.

TCP's ability to compensate for less than ideal network conditions is fundamentally affected by the RTT and MTU of the end-to-end Internet path that it traverses. The end-to-end path determines fixed bounds on these parameters. The target values for these three parameters, Data Rate, RTT and MTU, are determined by the application, its intended use and the physical infrastructure over which it is intended to traverse. These parameters are used to inform the models used to design the TDS.

This document describes a framework for deriving the traffic and delivery statistics for model based metrics. It does not fully specify any measurement techniques. Important details such as packet type-p selection, sampling techniques, vantage selection, etc are out

of scope for this document. We imagine Fully Specified Targeted Diagnostic Suites (FSTDS), that fully defines all of these details. We use TDS to refer to the subset of such a specification that is in scope for this document. A TDS includes specification for the traffic and delivery statistics for the diagnostic tests themselves, documentation of the models and any assumptions or derating used to derive the test parameters and a description of the test setup used to calibrate the models, as described in later sections.

Section 2 defines terminology used throughout this document.

It has been difficult to develop BTC metrics due to some overlooked requirements described in Section 3 and some intrinsic problems with using protocols for measurement, described in Section 4.

In Section 5 we describe the models and common parameters used to derive the targeted diagnostic suite. In Section 6 we describe common testing procedures. Each subpath is evaluated using suite of far simpler and more predictable diagnostic tests described in Section 7. In Section 8 we present three example TDS, one that might be representative of HD video, when served fairly close to the user, a second that might be representative of standard video, served from a greater distance, and a third that might be representative of an network designed to support high performance bulk download.

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

Future document will define model based metrics for other traffic classes and application types, such as real time streaming media.

1.1. TODO

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

Formatted: Mon Oct 21 15:42:35 PDT 2013

2. Terminology

Terminology about paths, etc. See [RFC2330] and [I-D.morton-ippm-lmap-path].

[data] sender Host sending data and receiving ACKs, typically via TCP.

[data] receiver Host receiving data and sending ACKs, typically via TCP.

subpath A portion of the full path. Note that there is no requirement that subpaths be non-overlapping.

Measurement Point Measurement points as described in [I-D.morton-ippm-lmap-path].

test path A path between two measurement points that includes a subpath of the end-to-end path under test, plus possibly additional infrastructure between the measurement points and the subpath.

[Dominant] Bottleneck The Bottleneck that determines a flow's self clock. It generally determines the traffic statistics for the entire path. See Section 4.1.

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

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

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

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

Properties determined by the end-to-end path and application. They are described in more detail in Section 5.1.

Application Data Rate General term for the data rate as seen by the application above the transport layer. This is the payload data rate, and excludes TCP/IP (or other protocol) headers and retransmits.

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

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

Target Data Rate: The application or ultimate user's performance goal. When converted to link data rate, it must be slightly smaller than the actual link data rate, otherwise there is no margin for compensating for RTT or other path properties. These test will be excessively brittle if the target data rate does not include any built in headroom.

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

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

Effective Bottleneck Data Rate: This is the bottleneck data rate that might be inferred from the ACK stream, by looking at how much data the ACK stream reports was delivered per unit time. See Section 4.1 for more details.

[sender] [interface] rate: The burst data rate, constrained by the data sender's interfaces. Today 1 or 10 Gb/s are typical.

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

Basic parameters common to models and subpath tests. They are described in more detail in Section 5.2.

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

target_pipe_size: The number of packets in flight (the window size) needed to exactly meet the target rate, with a single stream and no cross traffic for the specified target data rate, RTT and MTU.

run length A general term for the observed, measured or specified number of packets that are (to be) delivered between losses or ECN marks. Nominally one over the loss or ECN marking probability.

target_run_length Required run length computed from the target data rate, RTT and MTU.

Ancillary parameters used for some tests

derating: Under some conditions the standard models are too conservative. The modeling framework permits some latitude in relaxing or derating some test parameters as described in Section 5.3 in exchange for a more stringent TDS validation procedures, described in Section 9.

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

test_path_RTT The RTT (using appropriate packet sizes) between two measurement points.

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

test_window The window necessary to meet the target_rate over a subpath. Typically test_window=target_data_rate*test_RTT/ target_MTU.

Tests can be classified into groups according to their applicability

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

Monitoring tests are design to capture the most important aspects of a capacity test, but without causing unreasonable ongoing load themselves. As such they may miss some details of the network performance, but can serve as a useful reduced cost proxy for a capacity test.

Engineering tests evaluate how network algorithms (such as AQM and channel allocation) interact with TCP style self clocked protocols and adaptive congestion control based on packet loss and ECN marks. These tests are likely to have complicated interactions with other traffic and under some conditions can be inversely sensitive to load. For example a test to verify that an AQM algorithm causes ECN marks or packet drops early enough to limit queue occupancy may experience a false pass results in the presence of bursty 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.

3. New requirements relative to RFC 2330

[Move this entire section to a future paper]

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

- 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 must be vantage point invariant over a significant range of measurement point choices (e.g., measurement points as described in [I-D.morton-ippm-lmap-path]), including off path measurement points. The only requirements on MP selection should be that the portion of the path that is not under test is effectively ideal (or is non ideal in calibratable ways) and the RTT between MPs is below some reasonable bound.
- Metrics must be repeatable by multiple parties. It must be possible for different parties to make the same measurement and observe the same results. In particular it is specifically important that both a consumer (or their delegate) and ISP be able to perform the same measurement and get the same result.

NB: All of the metric requirements in RFC 2330 should be reviewed and potentially revised. If such a document is opened soon enough, this entire section should be dropped.

4. Background

[Move to a future paper, abridge here,]

At the time the IPPM WG was chartered, sound Bulk Transport Capacity measurement was known to be beyond our capabilities. By hindsight it is now clear why it is such a hard problem:

- TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test.
- Congestion control is an equilibrium process, transport protocols change the network (raise loss probability and/or RTT) to conform to their behavior.
- TCP's ability to compensate for network flaws is directly proportional to the number of roundtrips per second (i.e. inversely proportional to the RTT). As a consequence a flawed link may pass a short RTT local test even though it fails when the path is extended by a perfect network to some larger RTT.
- TCP has a meta Heisenberg problem - Measurement and cross traffic interact in unknown and ill defined ways. The situation is actually worse than the traditional physics problem where you can at least estimate the relative momentum of the measurement and

measured particles. For network measurement you can not in general determine the relative "elasticity" of the measurement traffic and cross traffic, so you can not even gage the relative magnitude of their effects on each other.

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

Models are used to determine the actual test parameters (burst size, loss rate, etc) from the target parameters. The basic method is to use models to estimate specific network properties required to sustain a given transport flow (or set of flows), and using a suite of metrics to confirm that the network meets the required properties.

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

- The raw link rate is higher than the target data rate.
- The raw packet run length is larger than required by a suitable TCP performance model
- There is sufficient buffering at the dominant bottleneck to absorb a slowstart rate burst large enough to get the flow out of slowstart at a suitable window size.
- 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 other mechanisms.
- When there is a standing queue at a bottleneck for a shared media subpath, there are suitable bounds on how the data and ACKs interact, for example due to the channel arbitration mechanism.
- When there is a slowly rising standing queue at the bottleneck the onset of packet loss has to be at an appropriate point (time or queue depth) and progressive.

The tests to verify these condition are described in Section 7.

A singleton [RFC2330] measurement is a pass/fail evaluation of a given path or subpath at a given performance. Note that measurements to confirm that a link passes at one particular performance might not be be useful to predict if the link will pass at a different

performance.

A TDS does have several valuable properties, such as natural ways to define several different composition metrics [RFC5835].

[Add text on algebra on metrics (A-Frame from [RFC2330]) and tomography.] The Spatial Composition of fundamental IPPM metrics has been studied and standardized. For example, the algebra to combine empirical assessments of loss ratio to estimate complete path performance is described in section 5.1.5. of [RFC6049]. We intend to use this and other composition metrics as necessary.

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

4.1. TCP properties

[Move this entire section to a future paper]

TCP and SCTP are self clocked protocols. The dominant steady state behavior is to have an approximately fixed quantity of data and acknowledgements (ACKs) circulating in the network. The receiver reports arriving data by returning ACKs to the data sender, the data sender most frequently responds by sending exactly the same quantity of data back into the network. The 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 widow by sending slightly more or less data in response to each ACK. The fundamentally important property of this systems is that it is entirely self clocked: The data transmissions are a reflection of the ACKs that were delivered by the network, the ACKs are a reflection of the data arriving from the network.

A number of phenomena can cause bursts of data, even in idealized networks that are modeled as simple queueing systems.

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

Another source of bursts are application pauses. If the application pauses (stops reading or writing data) for some fraction of one RTT, state-of-the-art TCP to "catches up" to the earlier window size by sending a burst of data at the full sender interface rate. To fill such a network with a realistic application, the network has to be able to tolerate interface rate bursts from the data sender large enough to cover application pauses.

Note that if the bottleneck data rate is significantly slower than the rest of the path, the slowstart bursts will not cause significant queues anywhere else along the path; they primarily exercise the queue at the dominant bottleneck. Furthermore, although the interface rate bursts caused by the application are likely to be smaller than last burst of a slowstart, they are at a higher rate so they can exercise queues at arbitrary points along the "front path" from the data sender up to and including the queue at the bottleneck.

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

Altering the ACK stream generally has two consequences: raising the effective bottleneck data rate making slowstart burst at higher rates (possibly as high as the sender's interface rate) and effectively raising the RTT by the time that the ACKs were postponed. The first effect can be partially mitigated by reclocking ACKs once they are beyond the bottleneck on the return path to the sender, however this further raises the effective RTT. The most extreme example of this class of behaviors is a half duplex channel that is never released until the current end point has no pending traffic. Such environments cause self clocked protocols revert to extremely inefficient stop and wait behavior, where they send an entire window of data as a single burst, followed by the entire window of ACKs on the return path.

If a particular end-to-end path contains a link or device that alters the ACK stream, then the entire path from the sender up to the bottleneck must be tested at the burst parameters implied by the ACK scheduling algorithm. The most important parameter is the Effective Bottleneck Data Rate, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACKs (relying on the cumulative nature of seg.ack to permit discarding some ACKs) is implies an effectively infinite bottleneck data rate.

To verify that a path can meet the performance target, it is

necessary to independently confirm that the entire path can tolerate bursts in the dimensions that are likely to be induced by the application and any data or ACK scheduling anywhere in the path. Two common cases are the most important: slowstart bursts at twice the effective bottleneck data rate; and somewhat smaller sender interface rate bursts.

The slowstart rate bursts must be at least as least as large target_pipe_size packets and should be twice as large (so the peak queue occupancy at the dominant bottleneck would be approximately target_pipe_size).

There is no general model for how well the network needs to tolerate sender interface rate bursts. All existing TCP implementations send full sized full rate bursts under some typically uncommon conditions, such as application pauses that approximately match the RTT, or when ACKs are lost or thinned. Strawman: partial window bursts (some fraction of target_pipe_size) should be tolerated without significantly raising the loss probability. Full target_pipe_size bursts may slightly increase the loss probability. Interface rate bursts as large as twice target_pipe_size should not cause deterministic packet drops.

5. Common Models and Parameters

5.1. Target End-to-end parameters

The target end to end parameters are the target data rate, target RTT and target MTU as defined in Section 2 These parameters are determined by the needs of the application or the ultimate end user and the end-to-end Internet path over which the application is expected to operate. The target parameters are in units that make sense to the upper layer: payload bytes delivered to the application, above TCP. They exclude overheads associated with TCP and IP headers, retransmitts and other protocols (e.g. DNS). In addition, other end-to-end parameters include the effective bottleneck data rate, the sender interface data rate and the TCP/IP header sizes (overhead).

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

The number of concurrent connections is explicitly not a parameter to this model [unlike earlier drafts]. If a subpath requires multiple connections in order to meet the specified performance, that must be

stated explicitly and the procedure described in Section 6.1.4 applies.

5.2. Common Model Calculations

The most important derived parameter is target_pipe_size (in packets), which is the window size --- the number of packets needed exactly meet the target rate, with no cross traffic for the specified target RTT and MTU. It is given by:

target_pipe_size = target_rate * target_RTT / (target_MTU - header_overhead)

If the transport protocol (e.g. TCP) average window size is smaller than this, it will not meet the target rate.

The reference target_run_length, is a very conservative model for the minimum required spacing between losses or ECN marks. The reference target_run_length can derived as follows: assume the subpath_data_rate is infinitesimally larger than the target_data_rate plus the required header overheads. Then target_pipe_size also predicts the onset of queueing. If the transport protocol (e.g. TCP) has a window size that is larger than the target_pipe_size, the excess packets will raise the RTT, typically by forming a standing queue at the bottleneck.

Assume the transport protocol is using standard Reno style Additive Increase, Multiplicative Decrease congestion control [RFC5681] and the receiver is using standard delayed ACKs. With delayed ACKs there must be 2*target_pipe_size roundtrips between losses. Otherwise the multiplicative window reduction triggered by a loss would cause the network to be underfilled. We derive the number of packets between losses from the area under the AIMD sawtooth following [MSMO97]. They must be no more frequent than every 1 in (3/2)*target_pipe_size*(2*target_pipe_size) packets. This simplifies to:

target_run_length = 3*(target_pipe_size^2)

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

These two parameters, target_pipe_size and target_run_length, directly imply most of the individual parameters for the tests below. Target_pipe_size is the window size, the amount of circulating data required to meet the target data rate, and implies the scale of the bursts that the network might experience. Target_run_length is the amount of data required between losses or ECN marks standard for standard congestion control.

The individual parameters are for each diagnostic test is described below. In a few case there are not well established models for what is considered correct network operation. In many of these cases the problems might either be partially mitigated by future improvements to TCP implementations.

5.3. Parameter Derating

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

- The TDS or FSTDS MUST document and justify the actual method used compute the derated metric parameters.
- The validation procedures described in Section 9 must be used to demonstrate the feasibility of meeting the performance targets with infrastructure that infinitessimally passes the derated tests.
- The validation process itself must be documented is such a way that other researchers can duplicate the validation experiments.

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

6. Common testing procedures

6.1. Traffic generating techniques

6.1.1. Paced transmission

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

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

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

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

Repeated Slowstart bursts: Slowstart bursts are typically part of larger scale pattern of repeated bursts, such as sending target_pipe_size packets as slowstart bursts on a target_RTT headway (burst start to burst start). Such a stream has three different average rates, depending on the averaging time scale. At the finest time scale the average rate is the same as the sender interface rate, at a medium scale the average rate is twice the effective bottleneck link rate and at the longest time scales the average rate is the target data rate.

Note that if the effective bottleneck link rate is more than half of the sender interface rate, slowstart bursts become sender interface rate bursts.

6.1.2. Constant window pseudo CBR

Implement pseudo constant bit rate by running a standard protocol such as TCP with a fixed bound on the window size. The rate is only maintained in average over each RTT, and is subject to limitations of the transport protocol.

The bound on the window size is computed from the target_data_rate and the actual RTT of the test path.

If the transport protocol fails to maintain the test rate within prescribed data rates, the test MUST NOT be considered passing. If there is a signature of a network problem (e.g. the run length is too small) then the test can be considered to fail. Since packet loss and ECN marks are required to reduce the data rate for standard transport protocols, the test specification must include suitable allowances in the prescribed data rates. If there is not sufficient signature of a network problem, then failing to make the prescribed data rate must be considered inconclusive. Otherwise there are some cases where tester failures might cause false negative test results.

6.1.3. Scanned window pseudo CBR

Same as the above, except the window is scanned across a range of sizes designed to include two key events, the onset of queueing and the onset of packet loss or ECN marks. The window is scanned by incrementing it by one packet for every 2*target_pipe_size delivered packets. This mimics the additive increase phase of standard congestion avoidance and normally separates the the window increases by approximately twice the target_RTT.

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

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

The test procedures in Section 7.2 describe how to the partition the scans into regions and how to interpret the results.

6.1.4. Concurrent or channelized testing

The procedures described in his document are only directly applicable to single stream performance measurement, e.g. one TCP connection. In an Ideal world, we would disallow all performance claims based multiple concurrent stream but this is not practical due to at least two different issues. First, many very high rate link technologies are channelized, and pin individual flows to specific channels to minimize reordering or solve other problems and second TCP itself has scaling limits. Although the former problem might be overcome through different design decisions, the later problem is more deeply rooted.

All standard [RFC 5681] and de facto standard [CUBIC] congestion control algorithms have scaling limits, in the sense that as a network over a fixed RTT and MTU gets faster all congestion control algorithms get less accurate. In general their noise immunity drops (a single packet drop should have less effect as individual packets

become smaller relative to the window size) and the control frequency of the AIMD sawtooth also drops, meaning that as TCP is using more total capacity it gets less information about the state of the network and other traffic. These properties are a direct consequence of the original Reno design and are implicitly required by the requirement that all transport protocols be "TCP friendly" [Guidelines] There are a number of reason to want to specify performance in term of multiple concurrent flows. Although there are a number of downsides to @@@@

The use of multiple connections in the Internet has been very controversial since the beginning of the World-Wide-Web[first complaint]. Modern browsers open many connections [BScope]. Experts associated with IETF transport area have frequently spoken against this practice [long list]. It is not inappropriate to assume some small number of concurrent connections (e.g. 4 or 6), to compensate for limitation in TCP. However, choosing too large a number is at risk of being interpreted as a signal by the web browser community that this practice has been embraced by the Internet service provider community. It may not be desirable to send such a signal.

Note that the current proposal for httpbis [SPDY] is specifically designed to work best with a single TCP connection per client server pair, because it uses adaptive compression which requires sending separate compression dictionaries per connection. As long as TCP can use IW10 and some of the transport parameter can be cached, multiple connections provide a negative gain, due to the replicated compression overhead.

The specification to use multiple connections is not recommended for data rates below several Mb/s, which can be attained with run lengths under 10000. Since run length goes as the square of the data rates, at higher rates (see Section 8.3) the run lengths can be unfeasibly large, and multiple connection might be the only feasible approach.

6.1.5. Intermittent Testing

Any test which does not depend on queueing (e.g. the CBR tests) or experiences periodic zero outstanding data during normal operation (e.g. between bursts for the various burst tests), can be formulated as an intermittent test.

The Intermittent testing can be used for ongoing monitoring for changes in subpath quality with minimal disruption users. It should be used in conjunction with the full rate test because this method assesses an average_run_length over a long time interval w.r.t. user sessions. It may false fail due to other legitimate congestion causing traffic or may false pass changes in underlying link

properties (e.g. a modem retraining to an out of contract lower rate).

[Need text about bias (false pass) in the shadow of loss caused by excessive bursts]

6.1.6. Intermittent Scatter Testing

Intermittent scatter testing: when testing the network path to or from an ISP subscriber aggregation point (CMTS, DSLAM, etc), intermittent tests can be spread across a pool of users such that no one users experiences the full impact of the testing, even though the traffic to or from the ISP subscriber aggregation point is sustained at full rate.

6.2. Interpreting the Results

6.2.1. Test outcomes

A singleton is a pass/fail measurement of a subpath. If any subpath fails any test then the end-to-end path is also expected to fail to attain the target performance under some conditions.

In addition we use "inconclusive outcome" to indicate that a test failed to attain the required test conditions. A test is inconclusive if the precomputed traffic pattern was not authentically generated, test preconditions were not met or the measurement results were not statistically significantly.

This is important to the extent that the diagnostic tests use protocols which themselves include built in control systems which might interfere with some aspect of the test. For example consider a test that is implemented by adding rate controls and loss instrumentation to TCP: meeting the run length specification while failing to attain the specified data rate must be treated as an inconclusive result, because we can not a priori determine if the reduced data rate was caused by a TCP problem or a network problem, or if the reduced data rate had a material effect on the run length measurement. (Note that if the measured run length was too small, 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 vantage independence properties of Model Based Metrics depends on the accuracy of the distinction between conclusive (pass or fail) and inconclusive tests. One way to view inconclusive tests is that they reflect situations where the signature is ambiguous between problems with the the subpath and problems with the diagnostic test itself. One of the goals for evolving diagnostic test designs will be to keep

sharpening this distinction.

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

Note that procedures that attempt to sweep the target parameter space to find the bounds on some parameter (for example to find the highest data rate for a subpath) are likely to break the location independent properties of Model Based Metrics, because the boundary between passing and inconclusive is extremely likely to be RTT sensitive, because TCP's ability to compensate for problems scales with the number of round trips per second.

6.2.2. Statistical criteria for measuring run_length

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

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

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

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

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

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

We also have an alternative Hypothesis to evaluate: if performance is significantly lower than the target_defect_probability. Based on analysis of typical values and practical limits on measurement

duration, we choose four times the H0 probability:

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

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

H0 and H1 constitute the Success and Failure outcomes described elsewhere in the memo, and while the ongoing measurements do not support either hypothesis the current status of measurements is inconclusive.

The problem above is formulated to match the Sequential Probability Ratio Test (SPRT) [StatQC], which also starts with a pair of hypothesis specified as above:

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

Acceptance line: Xa = -h1 + sn
Rejection line: Xr = h2 + sn
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 error.

The SPRT specifies simple stopping rules:

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

The calculations above are implemented in the R-tool for Statistical Analysis, in the add-on package for Cross-Validation via Sequential Testing (CVST) [http://www.r-project.org/] [Rtool] [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 + sn
and n = h1 / s

6.2.3. Reordering Tolerance

All tests must be instrumented for reordering [RFC4737].

NB: there is no global consensus for how much reordering tolerance is appropriate or reasonable. ("None" is absolutely unreasonable.)

Section 5 of [RFC4737] proposed a metric that may be sufficient to designate isolated reordered packets as effectively lost, because TCP's retransmission response would be the same.

[As a strawman, we propose the following:] TCP should be able to adapt to reordering as long as the reordering extent is no more than the maximum of one half window or 1 mS, whichever is larger. Note that there is a fundamental tradeoff between tolerance to reordering and how quickly algorithms such as fast retransmit can repair losses. Within this limit on reorder extent, there should be no bound on reordering density.

NB: Traditional TCP implementations were not compatible with this metric, however newer implementations still need to be evaluated

Parameters:
Reordering displacement: the maximum of one half of target_pipe_size or 1 mS.

6.3. Test Qualifications

This entire section might be summarized as "needs to be specified in a FSTDS"

Things to monitor before, during and after a test.

6.3.1. Verify the Traffic Generation Accuracy

[Excess detail for this doc. To be summarized]

for most tests, failing to accurately generate the test traffic indicates an inconclusive tests, since it has to be presumed that the error in traffic generation might have affected the test outcome. To the extent that the network itself had an effect on the the traffic generation (e.g. in the standing queue tests) the possibility exists that allowing too large of error margin in the traffic generation might introduce feedback loops that comprise the vantage independents properties of these tests.

Parameters:
Maximum Data Rate Error The permitted amount that the test traffic can be different than specified for the current test. This is a symmetrical bound.
Maximum Data Rate Overage The permitted amount that the test traffic can be above than specified for the current test.
Maximum Data Rate Underage The permitted amount that the test traffic can be less than specified for the current test.

6.3.2. Verify the absence of cross traffic

[Excess detail for this doc. To be summarized]

The proper treatment of cross traffic is different for different subpaths. In general when testing infrastructure which is associated with only one subscriber, the test should be treated as inconclusive it that subscriber is active on the network. However, for shared infrastructure, the question at hand is likely to be testing if provider has sufficient total capacity. In such cases the presence of cross traffic due to other subscribers is explicitly part of the network conditions and its effects are explicitly part of the test.

@@@@ Need to distinguish between ISP managed sharing and unmanaged sharing. e.g. WiFi

Note that canceling tests due to load on subscriber lines may introduce sampling errors for testing other parts of the infrastructure. For this reason tests that are scheduled but not run due to load should be treated as a special case of "inconclusive".

Use a passive packet or SNMP monitoring to verify that the traffic volume on the subpath agrees with the traffic generated by a test. Ideally this should be performed before, during and after each test.

The goal is provide quality assurance on the overall measurement process, and specifically to detect the following measurement failure: a user observes unexpectedly poor application performance, the ISP observes that the access link is running at the rated capacity. Both fail to observe that the user's computer has been infected by a virus which is spewing traffic as fast as it can.

Parameters:
Maximum Cross Traffic Data Rate The amount of excess traffic permitted. Note that this will be different for different tests.

One possible method is an adaptation of: www-didc.lbl.gov/papers/SCNM-PAM03.pdf D Agarwal etal. "An Infrastructure for Passive Network Monitoring of Application Data Streams". Use the same

technique as that paper to trigger the capture of SNMP statistics for the link.

6.3.3. Additional test preconditions

[Excess detail for this doc. To be summarized]

Send pre-load traffic as needed to activate radios with a sleep mode, or other "reactive network" elements (term defined in [draft-morton-ippm-2330-update-01]).

Use the procedure above to confirm that the pre-test background traffic is low enough.

7. Diagnostic Tests

The diagnostic tests are organized by which properties are being tested: run length, standing queues; slowstart bursts; sender rate bursts; and combined tests. The combined tests reduce overhead at the expense of conflating the signatures of multiple failures.

7.1. Basic Data Rate and Run Length Tests

We propose several versions of the basic data rate and run length test. All measure the number of packets delivered between losses or ECN marks, using a data stream that is rate controlled at or below the target_data_rate.

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

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

Pass: it is statistically significant that the observed run length is larger than the target_run_length.

Fail: it is statistically significant that the observed run length is smaller than the target_run_length.

A test is considered to be inconclusive if it failed to meet the data rate as specified below, meet the qualifications defined in

Section 6.3 or neither run length statistical hypothesis was confirmed in the allotted test duration.

7.1.1. Run Length at Paced Full Data Rate

Confirm that the observed run length is at least the target_run_length while relying on timer to send data at the target_rate using the procedure described in in Section 6.1.1 with a burst size of 1 (single packets).

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

7.1.2. run length at Full Data Windowed Rate

Confirm that the observed run length is at least the target_run_length while sending at an average rate equal to the target_data_rate, by controlling (or clamping) the window size of a conventional transport protocol to a fixed value computed from the properties of the test path, typically test_window=target_data_rate*test_RTT/target_MTU.

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

7.1.3. Background Run Length Tests

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

Existing loss metrics such as [RFC 6673] might be appropriate for measuring background run length.

7.2. Standing Queue tests

These test confirm that the bottleneck is well behaved across the onset of packet loss, which typically follows after the onset of

queueing. Well behaved generally means lossless for transient queues, but once the queue has been sustained for a sufficient period of time (or a sufficient queue depth) there should be a small number of losses to signal to the transport protocol that it should reduce its window. Losses that are too early can prevent the transport from averaging at the target_data_rate. Losses that are too late indicate that the queue might be subject to bufferbloat [Bufferbloat] and inflict excess queuing delays on all flows sharing the bottleneck. Excess losses make loss recovery problematic for the transport protocol. Non-linear or erratic RTT fluctuations suggest poor interactions between the channel acquisition systems and the transport self clock. All of the tests in this section use the same basic scanning algorithm but score the link on the basis of how well it avoids each of these problems.

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

Use the procedure in Section 6.1.3 to sweep the window across the onset of queueing and the onset of loss. The tests below all assume that the scan emulates standard additive increase and delayed ACK by incrementing the window by one packet for every 2*target_pipe_size packets delivered. A scan can be divided into three regions: below the onset of queueing, a standing queue, and at or beyond the onset of loss.

Below the onset of queueing the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the link rate, the data rate becomes fairly constant, and the RTT increases in proportion to the the window size. The precise transition from one region to the other can be identified by the maximum network power, defined to be the ratio data rate over the RTT[POWER].

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

If there is random background loss (e.g. bit errors, etc), precise determination of the onset of packet loss may require multiple scans. Above the onset of loss, all transport protocols are expected to experience periodic losses. For the stiffened transport case they will be determined by the AQM algorithm in the network or the details of how the the window increase function responds to loss. For the

standard transport case the details of periodic losses are typically dominated by the behavior of the transport protocol itself.

7.2.1. Congestion Avoidance

A link passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between the power point (or test_window) and the first loss or ECN mark. If this test is implemented using a standards congestion control algorithm with a clamp, it can be used in situ in the production internet as a capacity test. For an example of such a test see [NPAD].

7.2.2. Bufferbloat

This test confirms that there is some mechanism to limit buffer occupancy (e.g. prevents bufferbloat). Note that this is not strictly a requirement for single stream bulk performance, however if there is no mechanism to limit buffer occupancy then a single stream with sufficient data to deliver is likely to cause the problems described in [RFC 2309] and [Bufferbloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the link unusable for all other flows and applications.

Pass if the onset of loss is before a standing queue has introduced more delay than than twice target_RTT, or other well defined 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. Note that in conjunction with the previous test, this test implies that the first loss should occur at a queueing delay which is between one and two times the target_RTT.

7.2.3. Non excessive loss

This test confirm that the onset of loss is not excessive. Pass if losses are bound by the the fluctuations in the cross traffic, such that transient load (bursts) do not cause dips in aggregate raw throughput. e.g. pass as long as the losses are no more bursty than are expected from a simple drop tail queue. Although this test could be made more precise it is really included here for pedantic completeness.

7.2.4. Duplex Self Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path. Fail if the RTT rises by more than some fixed bound above the expected queueing time computed from trom the excess window divided by the link data rate. @@@@ This needs further testing.

7.3. Slowstart tests

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

They are deemed inconclusive if the elapsed time to send the data burst is not less than half of the time to receive the ACKs. (i.e. sending data too fast is ok, but sending it slower than twice the actual bottleneck rate as indicated by the ACKs is deemed inconclusive). Space the bursts such that the average data rate is equal to the target_data_rate.

7.3.1. Full Window slowstart test

This is a capacity test to confirm that slowstart is not likely to exit prematurely. Send slowstart bursts that are target_pipe_size total packets. Accumulate packet delivery statistics as described in Section 6.2.2 to score the outcome. Pass if it is statistically significant that the observed run length is larger than the target_run_length. Fail if it is statistically significant that the observed run length is smaller than the target_run_length.

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

7.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at slowstart_rate), until the first loss, stop, allow the network to drain and repeat, gathering statistics on the last packet delivered before the loss, the loss pattern, maximum RTT and window size. Justify the results. There is not currently sufficient theory justifying requiring any particular result, however design decisions that affect the outcome of this tests also affect how the network balances between long and short flows (the "mice and elephants" problem)

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

7.4. Sender Rate Burst tests

These tests determine how well the network can deliver bursts sent at sender's interface rate. Note that this test most heavily exercises the front path, and is likely to include infrastructure nominally out of scope.

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

Current standards permit TCP to send a full window bursts following an application pause. Congestion Window Validation [RFC 2861], is not required, but even if was it does not take effect until an application pause is longer than an RTO. Since this is standard behavior, it is desirable that the network be able to deliver it, otherwise application pauses will cause unwarranted losses.

It is also understood in the application and serving community that interface rate bursts have a cost to the network that has to be balanced against other costs in the servers themselves. For example TCP Segmentation Offload [TSO] reduces server CPU in exchange for larger network bursts, which increase the stress on network buffer memory.

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

For this reason, this is the only test for which we explicitly encourage detrateing. A TDS should include a table of pairs of derating parameters: what burst size to use as a fraction of the target_pipe_size, and how much each burst size is permitted to reduce the run length, relative to to the target_run_length. @@@@ Needs more work and experimentation.

7.5. Combined Tests

These tests are more efficient from a deployment/operational perspective, but may not be possible to diagnose if they fail.

7.5.1. Sustained burst test

Send target_pipe_size*derate sender interface rate bursts every target_RTT*derate, for derate between 0 and 1. Verify that the observed run length meets target_run_length. Key observations:

- This test is subpath RTT invariant, as long as the tester can generate the required pattern.
- The subpath under test is expected to go idle for some fraction of the time: (subpath_data_rate-target_rate)/subpath_data_rate. Failing to do so suggests a problem with the procedure.

- o This test is more strenuous than the slowstart tests: they are not needed if the link passes this test with derate=1.
- o A link that passes this test is likely to be able to sustain higher rates (close to subpath_data_rate) for paths with RTTs smaller than the target_RTT. Offsetting this performance underestimation is part of the rationale behind permitting derating in general.
- o This test can be implemented with standard instrumented TCP[RFC 4898], using a specialized measurement application at one end and a minimal service at the other end [RFC 863, RFC 864]. It may require tweaks to the TCP implementation.
- o This test is efficient to implement, since it does not require per-packet timers, and can make use of TSO in modern NIC hardware.
- o This test is not totally sufficient: the standing window engineering tests are also needed to be sure that the link is well behaved at and beyond the onset of congestion.
- o This one test can be proven to be the one capacity test to supplant them all.

7.5.2. Live Streaming Media

Model Based Metrics can be implemented as a side effect of serving any non-throughput maximizing traffic, such as streaming media, by applying some additional controls to the traffic. The essential requirement is that the traffic be constrained such that even with arbitrary application pauses, bursts and data rate fluctuations the traffic stays within the envelope determined by all of the individual tests described above, for a specific TDS.

If the serving RTT is less than the target_RTT, this constraint is most easily implemented by clamping the transport window size to test_window=target_data_rate*serving_RTT/target_MTU. This test_window size will limit the both the serving data rate and burst sizes to be no larger than the procedures in Section 7.1.2 and Section 7.4, assuming burst size derating equal to the serving_RTT divided by the target_RTT.

Note that if the application tolerates fluctuations in its actual data rate (say by use of a playout buffer) it is important that the target_data_rate be above the actual average rate needed by the application so it can recover after transient pauses caused by congestion or the application itself. Since the serving RTT is smaller than the target_RTT, the worst case bursts that might be generated under these conditions are smaller than called for by Section 7.4

8. Examples

In this section we present TDS for a couple of performance specifications.

Tentatively: 5 Mb/s*50 ms, 1 Mb/s*50ms, 250kbp*100mS

8.1. Near serving HD streaming video

Today the best quality HD video requires slightly less than 5 Mb/s [HDvideo]. Since it is desirable to serve such content locally, we assume that the content will be within 50 mS, which is enough to cover continental Europe or either US coast.

5 Mb/s over a 50 ms path

End to End Parameter	Value	units
target_rate	5	Mb/s
target_RTT	50	ms
traget_MTU	1500	bytes
target_pipe_size	22	packets
target_run_length	1452	packets

Table 1

This example uses the most conservative TCP model and no derating.

8.2. Far serving SD streaming video

Standard Quality video typically fits in 1 Mb/s [SDvideo]. This can be reasonably delivered via longer paths with larger. We assume 100mS.

5 Mb/s over a 50 ms path

End to End Parameter	Value	units
target_rate	1	Mb/s
target_RTT	100	ms
traget_MTU	1500	bytes
target_pipe_size	9	packets
target_run_length	243	packets

Table 2

This example uses the most conservative TCP model and no derating.

8.3. Bulk delivery of remote scientific data

This example corresponds to 100 Mb/s bulk scientific data over a moderately long RTT. Note that the target_run_length is infeasible for most networks.

100 Mb/s over a 200 ms path

End to End Parameter	Value	units
target_rate	100	Mb/s
target_RTT	200	ms
traget_MTU	1500	bytes
target_pipe_size	1741	packets
target_run_length	9093243	packets

Table 3

9. Validation

This document permits alternate models and parameter derating, as described in Section 5.2 and Section 5.3. In exchange for this latitude in the modelling process it requires the ability to demonstrate authentic applications and protocol implementations meeting the target end-to-end performance goals over infrastructure that infinitessimally passes the TDS.

The validation process relies on constructing a test network such that all of the individual load tests pass only infinitessimally, and

proving that an authentic application running over a real TCP implementation (or other protocol as appropriate) can be expected to meet the end-to-end target parameters on such a network.

For example using our example in our HD streaming video TDS described in Section 8.1, the bottleneck data rate should be 5 Mb/s, the per packet random background loss probability should be 1/1453, for a run length of 1452 packets, the bottleneck queue should be 22 packets and the front path should have just enough buffering to withstand 22 packet line rate bursts. We want every one of the TDS tests to fail if we slightly increase the relevant test parameter, so for example sending a 23 packet slowstart bursts should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. On this infinitessimally passing network it should be possible for a real ral application using a stock TCP implementation in the vendor's default configuration to attain 5 Mb/s over an 50 mS path.

@@@@ Need to better specify the workload: both short and long flows.

The difficult part of this process is arranging for each subpath to infinitesimally pass the individual tests. We suggest two approaches: constraining resources in devices by configuring them not to use all available buffer space or data rate; and preloading subpaths with cross traffic. Note that is it important that a single environment is constructed that infinitessimally 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.

If a TDS validated according to these procedures is used to inform public dialog, the validation experiment itself should also be public with sufficient precision for the experiment to be replicated by other researchers. All components should either be open source of fully specified proprietary implementations that are available to the research community.

TODO: paper proving the validation process.

10. Acknowledgements

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length.

Meredith Whittaker for improving the clarity of the communications.

11. Informative References

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

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

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

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

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

[I-D.morton-ippm-lmap-path] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for LMAP", draft-morton-ippm-lmap-path-00 (work in progress), January 2013.

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

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

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

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

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

[BScope] Broswerscope, "Browserscope Network tests", Sept 2012,

<http://www.browserscope.org/?category=network>.

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

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

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

[LMCUBIC] Ledesma Goyzueta, R. and Y. Chen, "A Deterministic Loss Model Based Analysis of CUBIC, IEEE International Conference on Computing, Networking and Communications (ICNC), E-ISBN : 978-1-4673-5286-4", January 2013.

Appendix A. Model Derivations

The reference target_run_length described in Section 5.2 is based on very conservative assumptions: that all window above target_pipe_size contributes to a standing queue that raises the RTT, and that classic Reno congestion control 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 standard, but the section provides offsetting requirements.

These models provide estimates that make the most sense if network performance is viewed logarithmically. In the operational internet, data rates span more than 8 orders of magnitude, RTT spans more than 3 orders of magnitude, and loss probability spans at least 8 orders of magnitude. When viewed logarithmically (as in decibels), these correspond to 80 dB of dynamic range. On an 80 db scale, a 3 dB error is less than 4% of the scale, even though it might represent a factor of 2 in raw parameter.

Although this document gives a lot of latitude for calculating target_run_length, people designing suites of tests need to consider the effect of their choices on the ongoing conversation and tussle about the relevance of "TCP friendliness" as an appropriate model for capacity allocation. Choosing a target_run_length that is substantially smaller than the reference target_run_length specified in Section 5.2 is equivalent to saying that it is appropriate for the transport research community to abandon "TCP friendliness" as a fairness model and to develop more aggressive Internet transport

protocols, and for applications to continue (or even increase) the number of connections that they open concurrently.

A.1. Aggregate Reno

In Section 5.2 it is assumed that the target rate is the same as the link rate, and any excess window causes a standing queue at the bottleneck. This might be representative of a non-shared access link. An alternative situation would be a heavily aggregated subpath where individual flows do not significantly contribute to the queueing delay, and losses are determined monitoring the average data rate, for example by the use of a virtual queue as in [AFD]. In such a scheme the RTT is constant and TCP's AIMD congestion control causes the data rate to fluctuate in a sawtooth. If the traffic is being controlled in a manner that is consistent with the metrics here, goal would be to make the actual average rate equal to the target_data_rate.

We can derive a model for Reno TCP and delayed ACK under the above set of assumptions: for some value of Wmin, the window will sweep from Wmin to 2*Wmin in 2*Wmin RTT. Between losses each sawtooth delivers (1/2)(Wmin+2*Wmin)(2Wmin) packets in 2*Wmin round trip times. However, unlike the queueing case where Wmin = Target_pipe_size, we want the average of Wmin and 2*Wmin to be the target_pipe_size, so the average rate is the target rate. Thus we want Wmin = (2/3)*target_pipe_size.

(@@@@ something is wrong above) Substituting these together we get:

target_run_length = (8/3)(target_pipe_size^2)

Note that this is always 88% of the reference run length.

A.2. CUBIC

CUBIC has three operating regions. The model for the expected value of window size derived in [LMCUBIC] assumes operation in the "concave" region only, which is a non-TCP friendly region for long-lived flows. The authors make the following assumptions: packet loss probability, p, is independent and periodic, losses occur one at a time, and they are true losses due to tail drop or corruption. This definition of p aligns very well with our definition of target_run_length and the requirement for progressive loss (AQM).

Although CUBIC window increase depends on continuous time, the authors transform the time to reach the maximum Window size in terms of RTT and a parameter for the multiplicative rate decrease on observing loss, beta (whose default value is 0.2 in CUBIC). The

expected value of Window size, E[W], is also dependent on C, a parameter of CUBIC that determines its window-growth aggressiveness (values from 0.01 to 4).

E[W] = (C*(RTT/p)^3 * ((4-beta)/beta))^-4

and, further assuming Poisson arrival, the mean throughput, x, is

x = E[W]/RTT

We note that under these conditions (deterministic single losses), the value of E[W] is always greater than 0.8 of the maximum window size ~= reference_run_length. (as far as I can tell)

Commentary on the consequence of the choice.

Appendix B. Version Control

Formatted: Mon Oct 21 15:42:35 PDT 2013

Authors' Addresses

Matt Mathis
Google, Inc
1600 Amphitheater Parkway
Mountain View, California 93117
USA

Email: mattmathis@google.com

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

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