IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: August 29, 2013

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

Feb 25, 2013

Model Based Internet Performance Metrics
draft-mathis-ippm-model-based-metrics-01.txt

Abstract

We introduce a new class of model based metrics designed to determine if a long path can meet predefined end-to-end application performance targets. This is done by section-by-section testing -- by applying a suite of single property tests to successive sections of a long path. In many cases these single property tests are based on existing IPPM metrics, with the addition of success and validity criteria. The sub-path at a time tests are designed to facilitate IP providers eliminating all known conditions that might prevent the full end-to-end path from meeting the users target 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 but are precomputed independently of the actual behavior of TCP over the sub-path under test. This makes the measurements open loop, eliminating nearly all of the difficulties encountered by traditional bulk transport metrics, which rely on congestion control equilibrium behavior.

A natural consequence of this methodology is verifiable network measurement: measurements from any given vantage point are repeatable from other vantage points.

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

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Table of Contents

1. Introduction 5
2. New requirements relative to RFC 2330 7
3. Background 7
4. Common Models and Parameters 9
4.1. Target End-to-end parameters 9
4.2. End-to-end parameters from sub-paths 11
4.3. Per sub-path parameters 11
4.4. Common Calculations for Single Property Tests 11
4.5. Parameter Derating 13
5. Common testing procedures 13
5.1. Traffic generating techniques 13
5.1.1. Paced transmission 14
5.1.2. Constant window pseudo CBR 14
5.1.3. Scanned window pseudo CBR 14
5.1.4. Intermittent Testing 15
5.1.5. Intermittent Scatter Testing 15
5.2. Interpreting the Results 15
5.2.1. Inconclusive test outcomes 15
5.2.2. Statistical criteria for measuring run_length 16
5.2.3. Classifications of tests 17
5.3. Reordering Tolerance 18
5.4. Verify the absence of cross traffic 18
5.5. Additional test preconditions 19
6. Single Property Tests 20
6.1. CBR Tests 21
6.1.1. Loss Rate at Full Data Rate 21
6.1.2. Background Loss Rate Tests 21
6.2. Standing Queue tests 22
6.2.1. Congestion Avoidance 22
6.2.2. Buffer Bloat 23
6.2.3. Self Interference 23
6.3. Slow Start tests 23
6.3.1. Full Window slow start test 23
6.3.2. Slowstart AQM test 24
6.4. Server Rate tests 24
6.4.1. Server TCP Send Offload (TSO) tests 24
6.4.2. Server Full Window test 24
7. Combined Tests 24
7.1. Sustained burst test 25
8. Calibration 26
9. Acknowledgements 26
10. References 26
10.1. Normative References 26
10.2. Informative References 26
Appendix A. Model Derivations 27
Appendix B. Old text from an earlier document 27

Authors' Addresses 29

1. Introduction

We introduce a new class of model based metrics designed to determine if a long path can be expected to meet a predefined application end-to-end performance target by applying a suite of single property tests to successive sections of the long path. In many cases these single property tests are based on existing IPPM metrics, with the addition of specific success and validity criteria. The sub-path at a time tests are designed to eliminate all known conditions that will potentially prevent the full path from meeting the target performance.

The end-to-end target performance must be specified in advance, and models are used to compute the IP layer properties necessary to support that performance. The IP metrics are based on traffic patterns that mimic TCP but are precomputed independently of the actual behavior of TCP over the sub-path under test.

This approach makes the measurements open loop, eliminating nearly all of the difficulties encountered by traditional bulk transport metrics, which depend on congestion control equilibrium behavior. Otherwise these control systems inherently have a number of properties that interfere with measurement: they have circular dependencies such that every component affects every property.

Since a singleton (see [RFC2330]) is only a pass/fail measurement of a sub-path, these metrics are most useful in composition over large pools of samples, such as across a collection of paths or a time interval [RFC5835] [RFC6049] .

For Bulk Transport Capacity (BTC) the target performance to be confirmed is a data rate. 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. Since the minimum RTT and maximum MTU are both fixed properties of the path, they are also taken as parameters to the modeling process. 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 traverses.

For BTC the following tests are sufficient:

- raw data rate,
- background loss rate,
- queue burst capacity,
- reordering extent [RFC4737],
- onset of congestion/AQM

- and corresponding metrics on return path quality.

If every sub-path passes all of these tests, then an end-to-end application using any reasonably modern TCP or similar protocol should be able to attain the specified target data rate, over the full end-to-end path at the specified RTT and MTU.

Traditional end-to-end BTC metrics have proven to be difficult or unsatisfactory due to some overlooked requirements described in Section 2 and some intrinsic difficulties with using protocols for measurement described in Section 3. In Section 4 we describe the models and common parameters used to derive single property test parameters. In Section 5 we describe common testing procedures used by all of the tests. Rather than testing the end-to-end path with TCP or other some other BTC, each sub-path is evaluated using suite of far simpler and more predictable single property tests described in Section 6. Section 7 describes some combined tests that are more efficient to implement and deploy. However, if they fail they may not clearly indicate the nature of the problem.

There exists the potential that model based metric itself might yield a false pass result, in the sense that every sub-path of an end-to-end path passes every single property test and yet a real application might still fall to attain its performance target over the path. If so, then a traditional BTC needs to be used to validate the tests for each sub-path, as described in Section 8.

Future text (or a more likely a future document) will describe model based metrics for real time traffic. The salient point will be that concurrently meeting the goals of both RT and throughput maximizing traffic implicitly requires some form of traffic segregation, such that the two traffic classes are not placed in the same queue. Some technique as simple as Fair Queueing [SFQ] might be a sufficient alternative to full QoS.

TODO:

- Better terminology for: single property test, test targets (as opposed to end-to-end targets), packet layer properties(?), testing suites, combined tests, etc. All to strengthen of the linguistic differences between transport and network layer.
- Make it clear that this document is about traffic patterns and delivery statistics. Other aspects of the test procedures are out of scope.
- Add description of assumed TCP behaviors used to derived the models.
- Eliminate sequentiality both as a modeling process and for section by section testing. Treatment of link under test being different from the bottleneck link (e.g. testing to an aggregation point).

- Add "effective bottleneck rate" as an end-to-end parameter. Discussion of ACK compression and its intrinsic consequences.
- Tests for a given subpath can be designed w/o knowing the rest of the path. Tests suites can be designed for link types in the abstract and standardized independently. Add example "complete test suites" following the combined tests.
- Add add concept of untargeted tests and algebra on loss rate.
- Make the background traffic test have an explicit procedure, and clearly delineate between users background traffic and other traffic. Connect preloading to intermittent and not intermittent, and as a way to control radio power states. Note that nearly all devices have some preloading effects (e.g. ARP on LANs)
- Better uniformity about: applies to all transport protocols, but defined in terms of TCP parameters.
- Clean and uniform descriptions of all tests.
- Write model appendix. Deprecate "old doc" appendix.

2. New requirements relative to RFC 2330

The Model Based Metrics are designed to fulfil some additional requirement that were not recognized at the time RFC 2330 was written. These missing requirements may have significantly contributed to policy difficulties in the IP measurement space. The 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 end-to-end 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 requirements for metrics in RFC 2330 should be reviewed and potentially revised.

3. Background

The holy grail of IPPM has been BTC measurement, but it has proven to

be a very hard problem for a number of reasons:

- 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 round trips per second (e.g. inversely proportional to the RTT). As a consequence a flawed link that passes a local test is likely to completely fail 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 masses of the measurement and measured particles. For network measurement you can not in general determine the relative "masses" of the measurement traffic and cross traffic, so you can not even gage the relative magnitude of their effects on each other.

The new approach is to "open loop" mandatory congestion control algorithms, typically by throttling TCP (or other protocol) to a lower rate, such that it does not react to changing network conditions. In this approack 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 almost entirely independent of the actual network behavior. 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 simpler metrics to confirm that the network meets the required properties. For example a network can sustain a Bulk TCP flow of a given data rate, MTU and RTT when 4 (and probably more) conditions are met:

- The raw link rate is higher than the target data rate.
- The raw packet loss rate is lower than required by a suitable TCP performance model
- There is sufficient buffering at any bottleneck smooth bursts.
- When the link is overfilled (congested), the onset of packet loss is progressive.

These condition can all be verified with simple tests, using model parameters and acceptance thresholds derived from the target data rate, MTU and RTT. Note that this procedure is not invertible: a singleton measurement is a pass/fail evaluation of a given path or subpath at a given performance. Measurements to confirm that a link passes at one particular performance may not be generally be useful to predict if the link will pass at a different performance.

Although they are not invertible, they do have several other valuable properties, such as natural ways to define several different composition metrics.

[Add text on algebra on metrics (A-Frame) 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.

4. Common Models and Parameters

Transport performance models are used to derive the test parameters for each single property test from the end-to-end target parameters and additional ancillary parameters.

It is envisioned that the modeling phase (to compute the test parameters) and testing phases will be decoupled. This section covers common derived parameters, used by multiple single property tests. For some tests, additional modeling is described with the tests. MAKE THIS NON SEQUENTIAL

Since some aspects of the models may be excessively conservative, the modeling framework permits some latitude in derating some test parameters, as described in Section 4.5.

For certain sub-paths (e.g. common types of access links) it would be appropriate for the single property test parameters to be documented as a "measurement profile" together with the modeling assumptions and derating factors described in Section 4.4 and Section 4.5.

4.1. Target End-to-end parameters

These parameters are determined by the needs of the application or the ultimate end user and the end-to-end Internet path. They are in units that make sense to the upper layer: payload bytes delivered, excluding header overheads for IP, TCP and other protocol.

Target Data Rate: The application or ultimate user's performance goal.

Target RTT (Round Trip Time): For fundamental reasons a long path makes it more difficult for TCP or other transport protocol to meet the target rate. The target RTT must be representative of the actual expected application use of the network. It may be subject to conventions about assumed application usage (e.g. continental scale paths should be assumed to be some fixed RTT, such as 100 ms) or alternatively be an property of an ISP's topology (e.g. a ISP with richer or better placed peering may be able to justify assuming lower RTTs than other ISPs.)

Target MTU (Maximum Transmission Unit): Assume 1500 Bytes per packet unless otherwise specified. If some sub-path forces a smaller MTU, then all sub-paths must be tested with the same smaller MTU.

Header overhead The IP and TCP header sizes, which are the portion of each MTU not available for carrying application payload. This is also assumed to be the size for returning acknowledgements (ACKs). The payload Maximum Segment Size (MSS) is the Target MTU minus header overhead.

Permitted Number of Connections: The target rate can be more easily obtained by dividing the traffic across more than one connection. (Ideally this would be 1). In general the number of concurrent connections is determined by the application, however see the comments below on multiple connections.

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. For traditional networks, the effective bottleneck rate would be the same as the actual bottleneck rate. If the forward path is subject AFD [AFD] style policing using a virtual queue, the effective bottleneck rate would be the same as the actual physical link rate, even though this rate is not available to the user. For systems that batch ACKs, for example due to half duplex channel allocation, the effective bottleneck data rate might be much higher than any link in the system.

The use of multiple connections 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.

4.2. End-to-end parameters from sub-paths

[This entire section needs to be overhauled and should be skipped on a first reading. The concepts defined here are not used elsewhere.]

The following optional parameters apply for testing generalized end-to-end paths that include subpaths with known specific types of behaviors that are not well represented by simple queueing models:

Bottleneck link clock rate: This applies to links that are using virtual queues or other techniques to police or shape users traffic at lower rates full link rate. The bottleneck link clock rate should be representative of queue drain times for short bursts of packets on an otherwise unloaded link.

Channel hold time: For channels that have relatively expensive channel arbitration algorithms, this is the typical (maximum?) time that data and or ACKs are held pending acquiring the channel. While under heavy load, the RTT may be inflated by this parameter, unless it is built into the target RTT

Preload traffic volume: If the user's traffic is shaped on the basis of average traffic volume, this is volume necessary to invoke "heavy hitter" policies.

Unloaded traffic volume: If the user's traffic is shaped on the basis of average traffic volume, this is the maximum traffic volume that a test can use and stay within a "light user" policies.

Note on a ConEx enabled network [ConEx], the word "traffic" in the last two items should be replaced by "congestion" i.e. "preload congestion volume" and "unloaded congestion volume".

4.3. Per sub-path parameters

[This entire section needs to be overhauled and should be skipped on a first reading. The concepts defined here are not used elsewhere.]

Some single parameter tests also need parameter of the sub-path.

sub-path RTT: RTT of the sub-path under test.

sub-path link clock rate: If different than the Bottleneck link clock rate

4.4. Common Calculations for Single Property Tests

The most important derived parameter is target_pipe_size (in packets), which is 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:

[Need to add multiple connections]

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, which is the most conservative model for the minimum spacing between losses, can derived as follows: assume the target_data_rate is equal to bottleneck link_data_rate. Then target_pipe_size also predicts the onset of queueing. If the transport protocol (e.g. TCP) has an average window size that is larger than the target_pipe_size, the excess packets will form a standing queue at the bottleneck.

If the transport protocol is using traditional Reno style Additive Increase, Multiplicative Decrease congestion control [RFC5681], then there must be target_pipe_size roundtrips between losses. Otherwise the multiplicative window reduction triggered by a loss would cause the network to be underfilled. Following [MSMO97], we derive the losses must be no more frequent than every 1 in (3/2)(target_pipe_size^2) packets. This provides the reference value for target_run_length which is typically the number of packets that must be delivered between loss episodes in the tests below:

reference_target_run_length = (3/2)(target_pipe_size^2)

Note that this calculation is based on a number of assumptions that may not apply. Appendix A discusses these assumptions and provides some alternative models. The actual method for computing target_run_length MUST be documented along with the rationale for the underlying assumptions and the ratio of chosen target_run_length to reference_target_run_length.

Although this document gives a lot of latitude for calculating target_run_length, people specifying profiles for suites of single property 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 reference_target_run_length 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.

The calculations for individual parameters are presented with the

each single property test. In general these calculations are permitted some derating as described in Section 4.5

4.5. Parameter Derating

Since some aspects of the models are very conservative, the modeling framework permits some latitude in derating some specific test parameters, as indicated in Section 6. For example classical performance models suggest that in order to be sure that a single TCP stream can fill a link, it needs to have a full bandwidth-delay-product worth of buffering at the bottleneck[QueueSize]. In real networks with real applications this is often overly conservative. 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 method used compute and justify the derated metrics is published in such a way that it becomes a matter of public record.
- The calibration procedures described in Section 8 are used to demonstrate the feasibility of meeting the performance targets with the derated test parameters.
- The calibration process itself is documented is such a way that other researchers can duplicate the experiments and validate the results.

In the test specifications in Section 6 assume 0 < derate <= 1, is a derating parameter. These will be individually named in the final document. In all cases making derate smaller makes the test more tolerant. Derate = 1 is "full strenght".

Note that some single property test parameters are not permitted to be derated.

5. Common testing procedures

5.1. Traffic generating techniques

A key property of Model Based Metrics is that the traffic patterns are determined by the end-to-end target parameters and not the network. The only exception are the constant window tests, which rely on a TCP style self clock. This makes the tests "open loop", which is key to preventing circular dependencies between test prameters. The transmission pattern does not depend on the details of the network's reaction to traffic.

5.1.1. Paced transmission

Paced (burst) transmissions: send bursts of data on a timer to meet a particular target rate and pattern.

Single: Send individual packets at the specified rate or headway.

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

Slow start: Send 4 packets bursts at an average rate equal to twice the effective bottleneck link rate (but not faster than the server interface rate). This corresponds to the average rate during a TCP slowstart when Appropriate Byte Counting [ABC] is present or delayed ack is disabled. Note that slow start pacing itself is typically part of larger scale burst patterns, such as sending target_pipe_size packets as slow start bursts every 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 server rate, at a medium scale the average rate is twice the bottleneck link rate and at the longest time scales the average rate is the target data rate, adjusted to include header overhead.

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

5.1.2. Constant window pseudo CBR

Implement pseudo CBR by running a standard protocol such as TCP at a fixed window size. This has the advantage that it can be implemented under real content delivery. The rate is only maintained in average over each RTT, and is subject to limitations of the transport protocol.

For tests that have strongly prescribed data rates, if the transport protocol fails to maintain the test rate for any reason (especially due to network congestion) the test should be considered inconclusive, otherwise there are some cases where tester failures might cause incorrect link tests results.

5.1.3. Scanned window pseudo CBR

Same as the above, except the window is incremented once per 2*target_RTT, starting from below target_pipe and sweeping up to first loss or some other event. This is analogous to the tests implemented in Windowed Ping [WPING] and pathdiag [PATHDIAG].

5.1.4. Intermittent Testing

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

The Intermittent testing can be used for ongoing monitoring for changes in sub-path 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).

5.1.5. Intermittent Scatter Testing

Intermittent scatter testing: when testing the network path to or from an ISP subscriber aggregation point (Cable headend [better word?] or DSLAM, etc), intermittent tests can be spread across a pool of (idle) 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. EXPAND

5.2. Interpreting the Results

(This section needs major reorganization.)

MOVE ELSEWHERE: General comment about types of loss: masking BER type loss with ARQ/FEC is not an issue. What we are most concerned with is congestion or AQM related losses. These are explicitly considered to be a signal back to the sender to slow down. Note that even with ARQ or FEC at some point the link will accumulate enough backlog where it will need to cause AQM (or drop tail overflow) losses.

5.2.1. Inconclusive test outcomes

A singleton is a pass fail measurement.

In addition we use "inconclusive" outcome to indicate that a test failed to attain the required test conditions. This is important to the extent that the tests themselves have built in control systems which might interfere with some aspect of the test.

For example if a test is implemented with an controled and instrumented TCP, failing to attain the specified data rate may indicate a problem with either the TCP implementation or the test vantage point.

One of the goal of the testing process should be to drive the number of inconclusive tests to zero.

5.2.2. Statistical criteria for measuring run_length

When evaluating the traget_run_length, we need to determine appropriate packet stream sizes and acceptable error levels to test efficiently. 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 a flight of packets and observe the packet transfer performance (loss ratio or other metric, any defect we define).

As each flight is sent and measured, we have an ongoing estimate of the performance in terms of defect to total packet ratio (or an empirical probability). 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 defects in target_run_length = (3/2)*(flight)^2 packets

and we can stop sending flights of packets if 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, say half the target:

H1: one or more defects in target_run_length/2 packets

and we can stop sending flights of packets if measurements support rejecting H0 with the specified Type II error = beta, thus preferring the alternate 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 indeterminate.

The problem above is formulated to match the Sequential Probability

Ratio Test (SPRT) [StatQC] [temp ref: http://en.wikipedia.org/wiki/Sequential_probability_ratio_test], which also starts with a pair of hypothesis specified as above:

H0: p = p0 = one defect in target_run_length
H1: p = p1 = one defect in target_run_length/2
As flights are sent and measurements collected, the tester evaluates the cumulative log-likelihood ratio:

S_i = S_i-1 + log(Lambda_i)

where Lambda_i is the ratio of the two likelihood functions (calculated on the measurement at packet i, and index i increases linearly over all flights of packets) for p0 and p1 [temp ref: http://en.wikipedia.org/wiki/Likelihood_function].

The SPRT specifies simple stopping rules:

- a < S_i < b: continue testing
- S_i <= a: Accept H0
- S_i >= b: Accept H1

where a and b are based on the Type I and II errors, alpha and beta:

a ~= Log((beta/(1-alpha)) and b ~= Log((1-beta)/alpha)

with the error probabilities decided beforehand, as above.

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] .

5.2.3. Classifications of tests

These tests are annotated with "(load)", "(engineering)" or "(monitoring)". WHY DO WE CARE?

A network would be expected to fail load tests in the presence excess or uncontrolled cross traffic. As such, load tests identify parameters that can transition from passing to failing as a consequence of insufficient network capacity and the actions of other network users.

Monitoring tests are design to capture the most important aspects of a load 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 load test.

Engineering tests evaluate how network algorithms (such as AQM and channel allocation) interact with transport protocols. Although these tests may be to be sensitive to load, the interaction may be quite complicated and might even have an inverted sensitivity. 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 under a full range of load conditions. Ongoing monitoring is less likely to be useful for these tests, although sparse testing might be appropriate.

5.3. Reordering Tolerance

All tests must be instrumented for excessive 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 extent, there should be no bound on the frequency.

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

5.4. Verify the absence of cross traffic

Cross traffic should be monitored prior to and during testing. In sub-paths where traffic of many users is aggregated, an excessive level of cross traffic should be noted and prevent testing (the test should be recorded as "inconclusive"). In sub-paths that include individual subscriber service, the current approach is to suspend testing when subscriber traffic is detected, because neither a flawed test nor degraded user performance conditions are desired.

Note that canceling tests due to load on subscriber lines may introduce sampling errors for 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".

The test is deemed to have passed only if the observed data rate matches the target_data_rate and it is statistically significant that the average_run_lenght is larger than target_run_lenght. It is deemed inconclusive if: the statistical test is inconclusive; there is too much background load; or the target_data_rate could not be attained.

Use a passive packet or SNMP monitoring to verify that the traffic volume on the sub-path 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 might be different for different tests.

Maximum Data Rate underage The permitted amount that the traffic can be less than predicted for the current test. Normally this would just be a statement of the maximum permitted measurement error, however it might also detect cases where the passive and active tests are misaligned: testing different subscriber lines. This is important because the vantage points are so different: in-band active measurement vs out-of-band passive measurement.

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.

5.5. Additional test preconditions

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.

6. Single Property Tests

The following tests are fully decomposed to verify individual network properties required for TCP meet the target parameters. It is believed that these properties apply to all self clocked throughput maximizing protocols. Failing to meet any one of these tests will cause poor TCP performance in some specific context.

These tests are pedantically separated: It would be more practical to combine them. Failing such a combined test might imply ambiguous consequences for TCP: it would be expected to fail under some conditions, but a single test might not be able to indicate exactly which conditions. The following section describes some combined tests.

The single property tests confirm that each sub-path can sustain the normal traffic patterns caused by TCP running at the specified target performance. Specifically they confirm that each sub-path has: sufficient raw capacity (e.g. sufficient data rate); low enough background loss rate where mandatory congestion control stays out of the way; large enough queue space to absorb TCP's normal bursts; does not cause unreasonable packet reordering; progressive AQM to appropriately invoke congestion control. Appropriately invoking congestion control requires that packet losses or ECN marks start progressively before TCP creates an excessive sustained queues[BufferBloat] or and without causing excessively bursty losses. The return path must also subject to a similar suite of tests, although potentially with different test parameters (due to the asymmetrical capabilities of many access link technologies). Furthermore it is important that the forward and return path interact appropriately, for example if they share they share a channel that has to be allocated.

Note that many of the sub-path tests resemble metrics that have already been defined in the IPPM context, with the specification of additional of test parameters and success critera. The models used to derive the test parameters make specific assumptions about network conditions, a test is deemed "inconclusive" (as opposed to failing) if tester does not meet the underlying assumption. For example a loss rate test at a specified data rate is inconclusive if the tester fails to send data at the specified rate for some reason. This concept of an inconclusive test is necessary to build tests out of protocols or technologies that they themselves have built in or implicit control systems.

6.1. CBR Tests

We propose several versions of the CBR loss rate test. One, performed at data full rate, is intrusive and recommend for infrequent testing, such as when a service is first turned up or as part of an auditing process. The second, background loss rate, is designed for ongoing monitoring for change is sub-path quality.

6.1.1. Loss Rate at Full Data Rate

Send at target_rate, confirm that the observed run length is at least the target_run_lenght. No additional derating is permitted (except through the choice of model to calculate target_run_lenght in the first place).

Note that this test also implicitly confirms that sub_path has sufficient capacity to carry the target_data_rate.

Parameters:
Run Length Same as target_run_lenght
Data Rate Same as target_data_rate

Note that these parameters MUST NOT be derated. If the default parameters are too stringent use an alternate model for target_run_lenght as described in Appendix A.

Data is sent at the specified data_rate. The receiver accumulates the total data delivered and packets lost [and ECN marks, which are nominally treated as losses by conforming transport protocols]. The observed average_run_lenght is computed from total_data_delivered divided by the total_loss_rate. A [TBD] statistical test is applied to determine when or if the average_run_length is larger than target_run_length.

TODO: add language about monitoring cross traffic. acm attempted below.

6.1.2. Background Loss Rate Tests

The background loss rate is a low rate version of the target rate test above, designed for ongoing monitoring for changes in sub-path quality without disrupting users. It should be used in conjunction with the above full rate test because it may be subject to false results under some conditions, in particular it may false pass changes in underlying link properties (e.g. a modem retraining to an out of contract lower rate).

Parameters:

Run Length Same as target_run_lenght
Data Rate Some small fraction of target_data_rate, such as 1%.

Once the preconditions described in Section 5.5 are met, the test data is sent at the prescribed rate with a burst size of 1. The receiver accumulates packet delivery statistics and the procedures described in Section 5.2.1 and Section 5.4 are used to score the outcome:

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

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

Inconclusive: Neither test was statistically significant or there was excess cross traffic during the test.

6.2. Standing Queue tests

These test confirm that the bottleneck is well behaved across the onset of queueing. For conventional bottlenecks this will be from the onset of queuing to the point where there is a full target_pipe of standing data. Well behaved generally means lossless for target_run_length, followed by a small number of losses to signal to the transport protocol that it should slow down. Losses that are too early can prevent the transport from averaging above the target_rate. Losses that are too late indicate that the queue might be subject to bufferbloat and subject other flows to excess queuing delay. Excess losses (more than half of of target_pipe) make loss recovery problematic for the transport protcol.

These tests can also observe some problems with channel acquisition systems, especially at the onset of persistent queueing. Details TBD.

6.2.1. Congestion Avoidance

Use the procedure in Section 5.1.3 to sweep the window (rate) from below link_pipe up to beyond target_pipe+link_pipe. Depending on events that happen during the scan, score the link. Identify the power_point=MAX(rate/RTT) as the start of the test.

Fail if first loss is too early (loss rate too high) on repeated tests or if the losses are more than half of the outstanding data. (a load test)

6.2.2. Buffer Bloat

Use the procedure in Section 5.1.3 to sweep the window (rate) from below link_pipe up to beyond target_pipe+link_pipe. Depending on events that happen during the scan, score the link. Identify the "power point:MAX(rate/RTT) as the start of the test (should be window=target_pipe)

Fail if first loss is too late (insufficient AQM and subject to bufferbloat - an engineering test). NO THEORY

6.2.3. Self Interference

Use the procedure in Section 5.1.3 to sweep the window (rate) from below link_pipe up to beyond target_pipe+link_pipe. Depending on events that happen during the scan, score the link. Identify the "power point:MAX(rate/RTT) as the start of the test (should be window=target_pipe)

Fail if RTT is non-monotonic by more than a small number of packet times (channel allocation self interference - engineering) IS THIS SUFFICIENT?

6.3. Slow Start tests

These tests mimic slow start: data is sent at twice subpath_rate. They are deemed inconclusive if the elapsed time to send the data burst is not less than half of the (extrapolated) time to receive the ACKs. (i.e. sending data too fast is ok, but sending it slower than twice the actual bottleneck rate is deemed inconclusive). Space the bursts such that the average ACK rate matches the target_data_rate.

These tests are not useful at burst sizes smaller than the server rate tests, since the server rate tests are more strenuous. If it is necessary to derate the server rate tests, then the full window slowstart test (un-derated) would be important.

6.3.1. Full Window slow start test

Send target_pipe_size*derate bursts must have fewer than one loss per target_run_length*derate. Note that these are the same parameters as the Server Full Window test, except the burst rate is at slowestart rate, rather than server interface rate. SHOULD derate=1.

Otherwise TCP will exit from slowstart prematurely, and only reach a full target_pipe_size window by way of congestion avoidance.

This is a load test: cross traffic may cause premature losses.

6.3.2. Slowstart AQM test

Do a continuous slowstart (date rate = 2*subpath_rate), until first loss, and repeat, gathering statistics on the last delivered packet's RTT and window size. Fail if too large (NO THEORY for value).

This is an engineering test: It would be best performed on a quiescent network or testbed, since cross traffic might cause a false pass.

6.4. Server Rate tests

These tests us "server interface rate" bursts. Although this is not well defined it should be assumed to be current state of the art server grade hardware (probably 10Gb/s today). (load)

6.4.1. Server TCP Send Offload (TSO) tests

If MIN(target_pipe_size, 42) packet bursts meet target_run_lenght (Not derated!).

Otherwise the link will interact badly with modern server NIC implementations, which as an optimization to reduce host side interactions (interrupts etc) accept up to 64kB super packets and send them as 42 seperate packets on the wire side.cc (load)

6.4.2. Server Full Window test

target_pipe_size*derate bursts have fewer than one loss per target_run_length*derate.

Otherwise application pauses will cause unwarranted losses. Current standards permit TCP to send a full cwnd burst following an application pause. (Cwnd validation in not required, but even so does not take effect until the pause is longer than RTO).

NB: there is no model here for what is good enough. derate=1 is safest, but may be unnecessarily conservative for some applications. Some application, such as streaming video need derate=1 to be efficient when the application pacing quanta is larger than cwnd. (load)

7. Combined Tests

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

7.1. Sustained burst test

Send target_pipe_size server rate bursts every target_RTT, verify that the observed run length meets target_run_length. Key observations:

- This test is 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: (link_rate-target_rate)/link_rate. Failing to do so suggests a problem with the procedure.
- This test is more strenuous than the slow start tests: they are not needed if the link passes underated server rate burst tests.
- This test could be derated by reducing both the burst size and headway (same average data rate).
- A link that passes this test is likely to be able to sustain higher rates (close to link_rate) for paths with RTTs smaller than the target_RTT. Offsetting this performance underestimation is the rationale behind permitting derating in general.
- This test should be implementable 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.
- This test is efficient to implement, since it does not require per-packet timers, and can make maximal use of TSO in modern NIC hardware.
- 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.
- I believe that this test can be proven to be the one load test to supplant them all.

Example

To confirm that a 100 Mb/s link can reliably deliver single 10 MByte/s stream at a distance of 50 mS, test the link by sending 346 packet bursts every 50 mS (10 MByte/s payload rate, assuming a 1500 Byte IP MTU and 52 Byte TCP/IP headers). These bursts are 4196288 bits on the wire (assuming 16 bytes of link overhead and framing) for an aggregate test data rate of 8.4 Mb/s.

To pass the test using the most conservative TCP model for a single stream the observed run length must be larger than 179574 packets. This is the same as less than one loss per 519 bursts (1.5*346) or every 26 seconds.

Note that this test potentially cause transient 346 packet queues at the bottleneck.

8. Calibration

If using derated metrics, or when something goes wrong, the results must be calibrated against a traditional BTC. The preferred diagnostic follow-up to calibration issues is to run open end-to-end measurements on an open platform, such as Measurement Lab [http://www.measurementlab.net/]

9. 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.

10. References

10.1. Normative References

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

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

10.2. 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.

[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.

[BScope] Broswerscope, "Browserscope Network tests", Sept 2012, <http://www.browserscope.org/?category=network>.

See Max Connections column

[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.

[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.

Appendix A. Model Derivations

This appendix describes several different ways to calculate target_run_length and the implication of the chosen calculation.

Rederive MSMO97 under two different assumptions: target_rate = link_rate and target_rate < 2 * link_rate.

Show equivalent derivation for CUBIC.

Commentary on the consequence of the choice.

Appendix B. Old text from an earlier document

To be moved, removed or absorbed

Step 0: select target end-to-end parameters: a target rate and target RTT. The primary test will be to confirm that the link quality is sufficient to meet the specified target rate for the link under test, when extended to the target RTT by an ideal network. The target rate must be below the actual link rate and nominally the target RTT would

be longer than the link RTT. There should probably be a convention for the relationship between link and target rates (e.g. 85%).

For example on a 10 Mb/s link, the target rate might be 1 MBytes/s, at an RTT of 100 mS (a typical continental scale path).

Step 1: On the basis of the target rate and RTT and your favorite TCP performance model, compute the "required run length", which is the required number of consecutive non-losses between loss episodes. The run length resembles one over the loss probability, if clustered losses only count as a single event. Also select "test duration" and "test rate". The latter would nominally the same as the target rate, but might be different in some situations. There must be documentation connecting the test rate, duration and required run length, to the target rate and RTT selected in step 0.

Continuing the above example: Assuming a 1500 Byte MTU. The calculated model loss rate for a single TCP stream is about 0.01% (1 loss in 1E4 packets).

Step 2, the actual measurement proceeds as follows: Start an unconstrained bulk data flow using any modern TCP (with large buffers and/or autotuning). During the first interval (no rate limits) observe the slowstart (e.g. tcpdump) and measure: Peak burst size; link clock rate (delivery rate for each round); peak data rate for the fastest single RTT interval; fraction of segments lost at the end of slow start. After the flow has fully recovered from the slowstart (details not important) throttle the flow down to the test rate (by clamping cwnd or application pacing at the sender or receiver). While clamped to the test rate, observe the losses (run length) for the chosen test duration. The link passes the test if the slowstart ends with less than approximately 50% losses and no timeouts, the peak rate is at least the target rate, and the measured run length is better than the required run length. There will also need to be some ancillary metrics, for example to discard tests where the receiver closes the window, invalidating the slowstart test. [This needs to be separated into multiple subtests]

Optional step 3: In some cases it might make sense to compute an "extrapolated rate", which is the minimum of the observed peak rate, and the rate computed from the specified target RTT and the observed run length by using a suitable TCP performance model. The extrapolated rate should be annotated to indicate if it was run length or peak rate limited, since these have different predictive values.

Other issues:

If the link RTT is not substantially smaller than the target RTT and the actual run length is close to the target rate, a standards compliant TCP implementation might not be effective at accurately controlling the data rate. To be independent of the details of the TCP implementation, failing to control the rate has to be treated as a spoiled measurement, not a infrastructure failure. This can be overcome by "stiffening" TCP by using a non-standard congestion control algorithm. For example if the rate controlling by clamping cwnd then use "relentless TCP" style reductions on loss, and lock ssthresh to the cwnd clamp. Alternatively, implement an explicit rate controller for TCP. In either case the test must be abandoned (aborted) if the measured run length is substantially below the target run length.

If the test is run "in situ" in a production environment, there also needs to be baseline tests using alternate paths to confirm that there are no bottlenecks or congested links between the test end points and the link under test.

It might make sense to run multiple tests with different parameters, for example infrequent tests with test rate equal to the target rate, and more frequent, less disruptive tests with the same target rate but the test rate equal to 1% of the target rate. To observe the required run length, the low rate test would take 100 times longer to run.

Returning to the example: a full rate test would entail sending 690 pps (1 MByte/s) for several tens of seconds (e.g. 50k packets), and observing that the total loss rate is below 1:1e4. A less disruptive test might be to send at 6.9 pps for 100 times longer, and observing

Formatted: Mon Feb 25 15:01:45 PST 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/