IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: December 23, 2013

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

June 21, 2013

Model Based Bulk Performance Metrics
draft-ietf-ippm-model-based-metrics-00.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 subpath at a time testing -- by applying a suite of single property tests to successive subpaths 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 subpath 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 subpath 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.

Formatted: Fri Jun 21 18:23:29 PDT 2013

Status of this Memo

Table of Contents

1. Introduction

Model based bulk performance metrics evaluate an Internet paths ability to carry bulk data. TCP models are used to design a targeted diagnostic suite of IP performance tests which can be applied independently to each subpath of the full end-to-end path. The targeted diagnostic suites are 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 targeted diagnostic suite 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 statistical criteria for evaluating the IP diagnostics tests. The criteria for passing each test must be determined from the end-to-end target performance and independent of the RTT or other properties 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 significantly.

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 which are both fixed properties of the end-to-end path. 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. They are used to inform the models used to design the targeted diagnostic suite.

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 used by all of the tests. Each subpath is evaluated using suite of far simpler and more predictable single property tests described in Section 7. Section 8 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 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 calibration procedure described in Section 9 needs to be used to validate and potentially revise the models.

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

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: Fri Jun 21 18:23:29 PDT 2013

2. Terminology

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

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. This must be slightly smaller than the actual link rate, otherwise there is no margin for compensating for RTT or other path protperties.

Target RTT (Round Trip Time): The RTT over which the application must meet the target performance.

Target MTU (Maximum Transmission Unit): Assume 1500 Bytes per packet unless otherwise specified. If some subpath forces a smaller MTU, then it becomes the target MTU, and all subpaths must be tested with 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.

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

[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.

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 Subpath as defined in [RFC2330].

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).

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

@ @@@

pipe size The number of packets needed in flight (the window size) to exactly fill some network path or sub path. The 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.

subpath pipe size

run length Observed, measured or specified number of packets that are (to be) delivered between losses or ECN marks. Nominally one over the loss probability.

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

reference_target_run_length: One specific conservative estimate of the number of packets that must be delivered between loss episodes in most diagnostic tests.

derating: The modeling framework permits some latitude in derating some specific test parameters as described in Section 5.3.

Test types [These need work]

capacity tests: For "capacity tests" is required that as long as the test traffic is within the proper envelope for the target end-to-end performance, the average packet losses must be below the threshold computed by the model.

Engineering tests: Engineering tests verify that the subpath under test interacts well with TCP style self clocked protocols using adaptive congestion control based on packet loss and ECN marks. For example "AQM Tests" verify that when the presented load exceeds the capacity of the subpath, the subpath signals for the transport protocol to slow down, by appropriately ECN marking or dropping some of the packets. Note while that cross traffic is can cause capacity tests to fail, it has the potential to cause AQM tests to false pass, which is why AQM tests require separate test procedures.

3. New requirements relative to RFC 2330

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. Some additional requirements are:

- o Metrics must be actionable by the ISP - they have to be interpreted in terms of behaviors or properties at the IP or lower layers, that an ISP can test, repair and verify.
- o Metrics 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.
- o 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

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:

- o TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test.
- o Congestion control is an equilibrium process, transport protocols change the network (raise loss probability and/or RTT) to conform to their behavior.
- o 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.
- o 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 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:

- o The raw link rate is higher than the target data rate.
- o The raw packet loss rate is lower than required by a suitable TCP performance model
- o 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.
- o 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.
- o 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.
- o 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.

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 [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.

4.1. TCP properties

TCP and SCTP are self clocked protocols. The dominant steady state behavior is to have an approximately fixed quantity of data and acknowledgements (ACKs) circulating in the network. The receiver reports arriving data by returning ACKs to the data sender, the data sender 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 by sending twice as much data as was delivered to the receiver. 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. 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 the worst case application pause.

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 burst at the last RTT of 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 ACKs and data stream, generally to raise the efficiency of the channel allocation process when confronted with relatively widely spaced ACKs. These efficiency strategies are ubiquitous for wireless and other half duplex or broadcast media.

Altering the ACK stream generally has two consequences: raising the effective bottleneck 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 through the bottleneck on the return 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 sender has no pending traffic. Such environments intrinsically 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 algorithms. 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, Model Based Metrics need to independently confirm that the entire path can tolerate bursts of the dimensions that are likely to be induced by the application and any data or ACK scheduling. Two common cases are the most important: slowstart bursts of with more than the target_pipe_size data at twice the effective bottleneck data rate; and somewhat smaller sender interface rate bursts.

5. Common Models and Parameters

Transport performance models are used to derive the test parameters for test suites of simple diagnostics from the end-to-end target

parameters and additional ancillary 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. They are in units that make sense to the upper layer: payload bytes delivered, excluding header overheads for IP, TCP and other protocol.

Ancillary parameters include the effective bottleneck rate and the permitted number of connections (numb_cons).

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.

5.2. Common Model Calculations

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

target_pipe_size = (target_rate / numb_cons) * 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 link_data_rate is infinitesimally larger than the target_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 standard 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. @@@ MOVE

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 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 permit some derating as described in Section 5.3. For test parameters that can be derated and are proportional to target_pipe_size, it is recommended that the derating be specified relative to target_pipe_size calculations using numb_cons=1, although the derating may additionally be specified relative to the target_pipe_size common to other tests.

5.3. Parameter Derating

Since some aspects of the models are very conservative, the modeling framework permits some latitude in derating some specific test parameters. 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:

- o The method used compute and justify the derated metrics is published in such a way that it becomes a matter of public record. @@@ introduce earlier

- The calibration procedures described in Section 9 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 7 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 test parameters are not permitted to be derated.

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.

Single: 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: Send 4 packet sender interface rate bursts at an average rate equal to the minimum of twice effective bottleneck link rate or 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: Slowstart pacing itself is 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 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 sender interface rate, slowstart bursts become sender interface rate bursts.

6.1.2. Constant window pseudo CBR

Implement pseudo CBR by running a standard protocol such as TCP with a fixed window size. This has the advantage that it can be implemented as part of 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 related to the network itself, such as packet losses or congestion, the test should be considered inconclusive. Otherwise there are some cases where tester failures might cause false negative link test results.

6.1.2.1. Scanned window pseudo CBR

Same as the above, except the window is incremented once per 2*target_pipe_size, starting from below target_pipe[@@@ test 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]

6.1.3. 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 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]

6.1.4. 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. If any subpath fails any test it can be assumed that the end-to-end path will also 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. This is important to the extent that the tests themselves use protocols that have built in control systems which might interfere with some aspect of the test. For example consider a test is implemented by adding rate controls and instrumentation to TCP: failing to attain the specified data rate has to be treated an inconclusive, unless the test clearly fails (target_run_lenght is too small). This is because failing to reach the target rate is an ambiguous signature for problems with either the test procedure (a problem with the TCP implementation or the test path RTT is too long) or the subpath itself.

The vantage independence properties of Model Based Metrics depends on the accuracy of the distinction between failing and inconclusive tests. One of the goals of evolving test designs will be to keep sharpening the distinction between failing and inconclusive tests.

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

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 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 inconclusive.

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:

- o a < S_i < b: continue testing
- o S_i <= a: Accept H0
- o 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] .

6.2.3. Classifications of tests

Tests are annotated with "(capacity)", "(engineering)" or "(monitoring)". @@@@MOVE to definitions?

Capacity tests determine if a network subpath has sufficient capacity to deliver the target performance. 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 network resources (capacity and/or buffer space), and their 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 transport protocols. These tests are likely to have complicated interactions with other network traffic and 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 under a variety of load conditions. Ongoing monitoring is less likely to be useful for these tests, although sparse in situ testing might be appropriate.

@@@ Add single property vs combined tests here?

6.2.4. 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 frequency.

NB: Current TCP implementations are not compatible with this metric. We view this as bugs in current TCP implementations.

Parameters:

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

6.3. Test Qualifications

Things to monitor before, during and after a test.

6.3.1. Verify the Traffic Generation Accuracy

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

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.

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

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. Single Property Tests

7.1. Basic Data and Loss Rate Tests

We propose several versions of the loss rate test. All are rate controlled at or below the target_data_rate. The first, performed at constant full data 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 subpath quality.

7.1.1. Loss Rate at Paced Full Data Rate

Confirm that the observed run length is at least the target_run_lenght while sending at the target_rate. This test implicitly confirms that sub_path has sufficient raw capacity to carry the target_data_rate. This version of the loss rate test relies on timers to schedule data transmission at a true constant bit rate (CBR).

Test Parameters:
Run Length Same as target_run_lenght
Data Rate Same as target_data_rate
Maximum Cross Traffic A specified small fraction of
   target_data_rate.

Note that target_run_lenght and target_data_rate parameters MUST NOT be derated. If the default parameters are too stringent an alternate model as described in Appendix A can be used to compute target_run_lenght.

The test traffic is sent using the procedures in Section 6.1.1 at target_data_rate with a burst size of 1, subject to the qualifications in Section 6.3. The receiver accumulates packet delivery statistics as described in Section 6.2 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: The test failed to meet the qualifications defined in Section 6.3 or neither test was statistically significant.

7.1.2. Loss Rate at Full Data Windowed Rate

Confirm that the observed run length is at least the target_run_lenght while sending at the target_rate. This test implicitly confirms that sub_path has sufficient raw capacity to carry the target_data_rate. This version of the loss rate test relies on a fixed window to self clock data transmission into the network. This is more authentic.

Test Parameters:
Run Length Same as target_run_lenght
Data Rate Same as target_data_rate
Maximum Cross Traffic A specified small fraction of
   target_data_rate.

Note that target_run_lenght and target_data_rate parameters MUST NOT be derated. If the default parameters are too stringent an alternate model as described in Appendix A can be used to compute target_run_lenght.

The test traffic is sent using the procedures in Section 6.1.1 at target_data_rate with a burst size of 1, subject to the qualifications in Section 6.3. The receiver accumulates packet delivery statistics as described in Section 6.2 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: The test failed to meet the qualifications defined in Section 6.3 or neither test was statistically significant.

7.1.3. 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 subpath 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 6.3 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 6.2.1 and Section 6.3 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.

7.2. 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.

7.2.1. Congestion Avoidance

Use the procedure in Section 6.1.2.1 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

capacity test)

7.2.2. Buffer Bloat

Use the procedure in Section 6.1.2.1 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

7.2.3. Duplex Self Interference

Use the procedure in Section 6.1.2.1 to sweep the window (rate) from below link_pipe up to beyond target_pipe+required_queue. 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) @@@ add required_queue and power_point

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

7.3. Slowstart tests

These tests mimic slowstart: data is sent at slowstart_rate (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 is equal to or faster than the target_data_rate.

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

7.3.1. Full Window slowstart test

Send (target_pipe_size+required_queue)*derate bursts must have fewer than one loss per target_run_length*derate. 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. 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 capacity test: cross traffic may cause premature losses.

7.3.2. Slowstart AQM test

Do a continuous slowstart (date rate = slowstart_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.

7.4. Sender Rate Burst tests

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

7.4.1. Sender 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)

7.4.2. Sender Full Window burst 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)

8. Combined Tests

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

8.1. Sustained burst test

Send target_pipe_size sender interface 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 slowstart tests: they are not needed if the link passes underated sender interface 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.

- o 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.
- o 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.
- 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 I believe that this test can be proven to be the one capacity 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.

9. 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/]

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.

[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.

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

This entire section is contains scraps of text to be moved, removed or absorbed elsewhere in the document

B.1. An earlier document

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 slowstart. 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

B.2. End-to-end parameters from subpaths

[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".

B.3. Per subpath 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 subpath.

subpath RTT: RTT of the subpath under test.

subpath link clock rate: If different than the Bottleneck link clock rate

B.4. Version Control

Formatted: Fri Jun 21 18:23:29 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/