Network Working Group
Internet Draft
Intended status: Informational
Expires: June 2013
December 1, 2012

B. Constantine
JDSU
T. Copley
Level-3
R. Krishnan
Brocade Communications

# Traffic Management Benchmarking
draft-constantine-bmwg-traffic-management-00.txt

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups.  Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on July 5, 2013.

## Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Abstract

This framework describes a practical methodology for benchmarking the traffic management capabilities of networking devices (i.e. policing, shaping, etc.). The goal is to provide a repeatable test method that objectively compares performance of the device’s traffic management capabilities and to specify the means to benchmark traffic management with representative application traffic.

## Table of Contents

## 1. Introduction

Traffic management (i.e. policing, shaping, etc.) is an increasingly important component in today’s networks.  There is no framework to benchmark these features although some standards address specific areas.  This draft provides a framework to conduct repeatable traffic management benchmarks for devices and systems in a lab environment. The benchmarking framework can also be used as a test procedure to assist in the tuning of Quality of Service (QoS) parameters before field deployment.  In addition to Layer 2/3 benchmarking, techniques to define Layer 4 traffic test patterns are presented that can benchmark the traffic management technique(s) under realistic conditions.

### 1.1. Traffic Management Overview

In general, a device with traffic management capabilities performs the following QoS functions:

. Traffic classification: identifies traffic according to various QoS rules (i.e. VLAN, DSCP, etc.) and marks this traffic internally to the network device (for traffic management processing)

. Traffic policing: rate limits traffic that enters a router according to the traffic classification.  If the traffic exceeds the contracted Service Level Agreement (SLA), the traffic is either dropped or remarked and sent onto to the next network node

. Traffic shaping: is a traffic control measure of actively buffering and metering the output rate of traffic in an attempt to adapt bursty traffic to the SLA.

.  Traffic Scheduling: provides QoS within the network device by storing packets in various types of queues and applies a dispatching algorithm to assign the forwarding sequence of packets.

. Congestion Management: monitors the status of internal queues and actively drops packets, which causes the sending hosts to back-off and in turn can alleviate queue congestion.

The following diagram is a generic model of the traffic management capabilities within a network device.  It is not intended to represent all variations of manufacturer traffic management capabilities, but provide context to this test framework.

```
|----------|   |----------------|    |-------------|   |----------|
|          |   |                |    |             |   |          |
|Interface |   |Ingress QoS     |    |Egress QoS   |   |Interface |
|Input     |   |(classification,|    |(scheduling, |   |Output    |
|Queues    |   | marking,       |    |  shaping,   |   |Queues    |
|          |-->| policing or    |--->|  congestion |-->|          |
|          |   | shaping)       |    |  management |   |          |
|          |   |                |    |  re-marking)|   |          |
|          |   |                |    |             |   |          |
|----------|   |----------------|    |-------------|   |----------|
```

Figure 1: Generic Model of Traffic Management capabilities within a network device

(TC comment: A couple other things that a traffic management device must be able to perform Is Marking / Remarking / encapsulation.  I also think we should be looking at the performance that these types of functions add to the packet.)

## 2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

The following acronyms are used:

BDP: Bandwidth Delay Product

CBS: Committed Burst Size

CIR: Committed Information Rate

DUT: Device Under Test

EBS: Exceeded Burst Size

EIR: Exceeded Information Rate

QoS: Quality of Service

RED: Random Early Discard

RTT: Round Trip Time

WRED: Weighted Random Early Discard

The following is the description of the lab set-up for the traffic management tests:

```
 +--------------+     +-------+     +----------+    +-----------+
 | Transmitting |     |       |     |          |    | Receiving |
 | Test Host    |     |       |     |          |    | Test Host |
 |              |-----| DUT   |---->| Network  |--->|           |
 |              |     |       |     | Delay    |    |           |
 |              |     |       |     | Emulator |    |           |
 |              |<----|       |<----|          |<---|           |
 |              |     |       |     |          |    |           |
 +--------------+     +-------+     +----------+    +-----------+
```

As shown the test diagram, the framework supports uni-directional and bi-directional traffic management tests.

Also note that the Network Delay Emulator (NDE) should be passive in nature such as a fiber spool.  This is recommended to eliminate the potential effects that an active delay element (i.e. test impairment generator) may have on the test flows. In the case that a fiber spool is not practical due to the desired latency, an active NDE must be independently verified to be capable of adding the configured delay without loss.  This requirement will vary from test to test on desired traffic speed and should be calibrated before any test requiring delay, which can add a significant additional amount of testing to each step.

## 3. Scope and Goals

The scope of this work is to develop a framework for benchmarking and testing the traffic management capabilities of network devices in the lab environment.  These network devices may include but are not limited to:

- Switches (including Layer 2/3 devices)

- Routers

- Firewalls

Essentially, any network device that performs traffic management as defined in section 1.1 can be benchmarked or tested with this framework.

Within this framework, the metrics are defined for each traffic management test but do not include pass / fail criterion, which is not within the charter of BMWG.  This framework does not attempt to rate the performance of one manufacturer’s network equipment versus another, but only to provide benchmarks to conduct repeatable, comparative testing.

A goal of this framework is to define specific stateless traffic ("packet blasting") tests to conduct the benchmark tests and also to derive stateful test patterns (TCP or application layer) that can also be used to further benchmark the performance of applicable traffic management techniques such as traffic shaping and congestion management techniques such as RED/WRED.  In cases where the network device is stateful in nature (i.e. firewall, etc.), stateful test pattern traffic is the only option.

And finally, this framework will provide references to open source tools that can be used to provide the stateless traffic generation capabilities and the stateful emulation capabilities referenced above.

## 4. Traffic Benchmarking Metrics

The metrics to be measured during the benchmarks are divided into two (2) sections: packet layer metrics used for the stateless traffic testing and metrics used for the stateful traffic testing

### 4.1. Metrics for Stateless Traffic Tests

The following are the metrics to be used during the stateless traffic benchmarking components of the tests:

- Burst Size Achieved (BSA): for the traffic policing and network queue tests, the tester will be configured to send bursts to test either the Committed Burst Size (CBS) or Exceeded Burst Size (EBS) of a policer or the queue / buffer size configured in the DUT. The Burst Size Achieved metric is a measure of the actual burst size received at the egress port of the DUT with no lost frames. As an example, the CBS of a DUT is 64KB and after the burst test, only a 63 KB can be achieved without frame loss. Then 63KB is the BSA.

- Lost Frames (LF): For all traffic management tests, the tester will transmit the test frames into the DUT ingress port and the number of frames received at the egress port will be measured. The difference between frames transmitted into the ingress port and received at the egress port is the number of lost frames as measured at the egress port. These frames must have unique identifiers such that only the test frames are measured.

- Out of Sequence Frames (OOS): in additions to LF metric, the test frames must be monitored for sequence and the out-of-sequence (OOS) frames will be counted per RFC-???? or is this ITU??.

- Frame Delay (FD): the Frame Delay metric is the difference between the timestamp of the received egress port frames and the frames transmitted into the ingress port and specified in ITU-1564.

- Frame Delay Variation (FDV): the Frame Delay Variation metric is the variation between the timestamp of the received egress port frames and specified in ITU-1564.

(Note, we need to consider bi-directional nature of the tests and metrics)

## 4.2. Metrics for Stateful Traffic Tests

The stateful metrics will be based on RFC 6349 TCP metrics and will included the following:

- TCP Test Pattern Execution Time: RFC 6349 defined the TCP Transfer Time for bulk transfers, which is simply the measured time to transfer bytes across single or concurrent TCP connections.  The TCP test patterns used in traffic management tests will be bulk transfer and interactive in nature; these test patterns simulate delay-tolerant applications like FTP, streaming video etc..  The TTPET will be the measure of the time for a single execution of a TTPET. Average, minimum, and maximum times will be measured.

- TCP Efficiency: after the execution of the TCP Test Pattern, TCP Efficiency represents the percentage of Bytes that were not retransmitted.

```
                         Transmitted Bytes - Retransmitted Bytes

   TCP Efficiency % =  ---------------------------------------  X 100

                                  Transmitted Bytes
```

Transmitted Bytes are the total number of TCP Bytes to be transmitted including the original and the retransmitted Bytes.

- Buffer Delay: represents the increase in RTT during a TCP test versus the baseline DUT RTT (non congested, inherent latency).  The average RTT is derived from the total of all measured RTTs during the actual test at every second divided by the test duration in seconds.

```
                                        Total RTTs during transfer

      Average RTT during transfer = -----------------------------

                                       Transfer duration in seconds



                       Average RTT during Transfer - Baseline RTT

    Buffer Delay % = ------------------------------------------ X 100
```

Baseline RTT

## 5. Tester Capabilities

The testing capabilities of the traffic management test environment are divided into two (2) sections: stateless traffic testing and stateful traffic testing

### 5.1. Stateless Test Traffic Generation

The test set must be capable of generating test traffic at up to the link speed of the DUT. The test set must be calibrated to verify that it will not drop any frames. The test set’s inherent FD and FDV must also be calibrated and subtracted from the FD and FDV metrics.

The test set must support the encapsulation to be tested such as VLAN, Q-in-Q, MPLS, etc.

The open source tool "iperf" can be used to generate stateless UDP traffic and is discussed in Appendix A. Since iperf is a software based tool, there will be performance limitations at higher link speeds. Careful calibration of any test environment using iperf is important. At higher link speeds, it is recommended to select commercial hardware based packet test equipment.

### 5.2. Stateful Test Pattern Generation

The TCP test host will have many of the same attributes as the TCP test host defined in RFC 6349. The TCP test host may be a standard computer or a dedicated communications test instrument. In both cases, it must be capable of emulating both a client and a server.

For any test using stateful TCP test traffic, the Network Delay Emulator (NDE) function from the lab set-up must be used in order to provide a meaningful BDP. As referenced in section 2, the target traffic rate and configured RTT must be verified independently using just the NDE for all stateful tests (to ensure the NDE can delay without loss).

The TCP test host must be capable to generate and receive stateful TCP test traffic at the full link speed of the DUT. As a general rule of thumb, testing TCP Throughput at rates greater than 100 Mbps may require high performance server hardware or dedicated hardware based test tools.

(TC comment: You mention that a device to do rates greater than 100Mbit may require a high performance server.  We also need to discuss how window Sizes or flows impact that.)

The TCP test host must allow adjusting both Send and Receive Socket Buffer sizes.  The Socket Buffers must be large enough to fill the BDP for bulk transfer TCP test application traffic.

Measuring RTT and retransmissions per connection will generally require a dedicated communications test instrument. In the absence of dedicated hardware based test tools, these measurements may need to be conducted with packet capture tools, i.e. conduct TCP Throughput tests and analyze RTT and retransmissions in packet captures.

The TCP implementation used by the test host must be specified in the test results (i.e. OS version, i.e. LINUX OS kernel using TCP New Reno, TCP options supported, etc).

While RFC 6349 defined the means to conduct throughput tests of TCP bulk transfers, the traffic management framework will extend TCP test execution into interactive TCP application traffic.  Examples include email, HTTP, business applications, etc.  This interactive traffic is not uni-directional in nature but is chatty.

The test host must not only support bulk TCP transfer application traffic but this chatty traffic since the both stress traffic management techniques in very different ways.  This is due to the non-uniform, bursty nature of chatty applications versus the relatively uniform nature of bulk transfers (the bulk transfer smoothly stabilizes to equilibrium state under lossless conditions).

While iperf is an excellent choice for TCP bulk transfer testing, the open source tool "Flowgrind" is applicable to interactive TCP flows and is also referenced in Appendix A.  Flowgrind is client server based and emulates interactive applications at the TCP layer.  As with any software based tool, the performance must be qualified to the link speed to be tested.  Commercial test equipment should be considered for reliable results at higher links speeds.

### 5.2.1. TCP Test Pattern Definitions

As mentioned in the goals of this framework, techniques to define Layer 4 traffic test patterns will be defined to benchmark the traffic management technique(s) under realistic conditions.  Some network devices such as firewalls, will not process stateless test traffic which is another reason that stateful TCP test traffic must be used.

An application can be fully emulated to Layer 7 but this framework proposes that stateful TCP test patterns be used to provide granular and repeatable control for the benchmarks. The following diagram illustrates a simple Web Browsing application (HTTP).

```
                GET url

Client      ------------------------->   Web

Web             200 OK        100ms |

Browser     <-------------------------    Server
```

In this example, the Client Web Browser (Client) requests a URL and then the Web Server delivers the web page contents to the Client (after a Server delay of 100 msec).  This synchronous, "request / response" behavior is intrinsic to most TCP based applications such as Email (SMTP), File Transfers (FTP and SMB), Database (SQL), Web Applications (SOAP), etc.   The impact to the network elements is due to the multitudes of Clients and the variety of bursty traffic, which stresses network resources such as buffers, shapers, and other QoS management techniques.  The actual emulation of the specific application protocols is not required and TCP test patterns can be defined to mimic the application behavior.

This framework does not specify a fixed set of TCP test patterns, but does provide examples in Appendix B.  There are two (2) techniques recommended by this framework to develop standard TCP test patterns for traffic management benchmarking.

The first technique involves modeling techniques, which have been described in "3GPP2 C.R1002-0 v1.0" and describe the behavior of HTTP, FTP, and WAP applications at the TCP layer.  The models have been defined with various mathematical distributions for the Request/Response bytes and inter-request gap times.  The Flowgrind tool (Appendix A) supports many of the distributions and is a good choice as long as the processing limits of the server platform are taken into consideration.

The second technique is to conduct packet captures of the applications to test and then to statefully play the application back at the TCP layer.  The TCP playback includes the request byte size, response byte size, and inter-message gaps at both the client and the server.  The advantage of this method is that very realistic test patterns can be defined based off of real world application traffic.

Appendix B provides an overview of the modeling technique with Flowgrind, capture technique with TCP playback, and some representative application traffic that can be used with either technique.

(TC comment: In addition to application test patterns,  I'd also like to see some of the standard ways mentioned like 2544 all 1's all F's all 0's and the Alternating)

## 6. Traffic Benchmarking Methodology

The traffic benchmarking methodology uses the test set-up from section 2 and metrics defined in section 4.  Each test should be run for a minimum test time of 5 minutes.

### 6.1. Policing Tests

The intent of the policing tests is to verify the policer performance parameters of CIR-CBS and EIR-EBS. The tests will verify that the device can handle the CIR rate with CBS and the EIR rate with EBS and will use back-back frame testing concepts from RFC 2544 (but adapted to burst size algorithms and terminology).  Also MEF-14,19,37 provide some basis for specific components of this test.

Policing tests will only use stateless traffic since a policer only operates at Layer 2.  Stateful TCP test traffic would not yield any benefit to test a policer.

The policer test traffic shall follow the traffic profile as defined in MEF 10.2.  Specifically, the stateless traffic shall be transmitted at the link speed within the time interval of the policer.  In MEF 10.2, this time interval is defined as:

Tc = (CBS * 8) / CIR or

Te = (EBS * 8) / EIR

As an example, consider a CBS of 64KB and CIR of 100 Mbps on a 1GigE physical link. The Tc equates to 5.12 msec and the 64KB burst should be transmitted into the ingress port at full GigE rate, then wait for 5.12 msec for the next burst, etc.

The metrics defined in section 4.1 shall be measured at the egress port and recorded; the primary result is to verify the BSA and that no frames are dropped.

In addition to verifying that the policer allows the specified CBS and EBS bursts to pass, the policer test must verify that the policer will police at the specified CBS/EBS values.

For this portion of the test, the CBS/EBS value should be incremented by 1000 bytes higher than the configured CBS and that the egress port measurements must show that the majority of frames are dropped.

## 6.2. Queue Tests

The queue tests are similar in nature and can be covered with the same test technique for the stateless traffic tests. There are not CIR-CBS, EIR-EBS parameters for network device queues so only the CBS component of the policer tests should be applied to pure queue tests.

Since device queues / buffers are generally an egress function, this test framework will discuss testing at the egress (although the technique can be applied to ingress side queues).

### 6.2.1. Testing Queue with Stateless Traffic

A network device queue is memory based unlike a policing function, which is token or credit based. However, the same concepts from section 6.1 can be applied to testing network device queue.

The device’s network queue should be configured to the desired size in KB (queue length, QL) and then stateless traffic should be transmitted to test this QL.

The transmission interval (Ti) can be defined for the traffic bursts and is based off of the QL and Bottleneck Bandwidth (BB) of the egress interface. The equation is similar to the Tc / Te time interval discussed in the policer section 6.1 and is as follows:

Ti = QL * 8 / BB

Important to note that the assumption is that the aggregate ingress throughput is higher than the BB or the queue test is not relevant since there will not be any over subscription.

The stateless traffic shall be transmitted at the link speed within the Ti time interval. The metrics defined in section 4.1 shall be measured at the egress port and recorded; the primary result is to verify the BSA and that no frames are dropped.

### 6.2.2. Testing Queue with Stateful Traffic

To provide a more realistic benchmark and to test queues in layer 4 devices such as firewalls, stateful traffic testing is recommended for the queue tests. Stateful traffic tests will also utilize the Network Delay Emulator (NDE) from the network set-up configuration in section 2.

The BDP of the TCP test traffic must be calibrated to the QL of the device queue. The BDP is equal to:

BB * RTT / 8 (in bytes)

The NDE must be configured to an RTT value which is great enough to allow the BDP to be greater than QL. An example test scenario is defined below:

- Ingress link = GigE

- Egress link = 100 Mbps (BB)

- QL = 32KB

RTT(min) = QL * 8 / BB and would equal 2.56 msec and the BDP = 32KB

In this example, one (1) TCP connection with window size / SSB of 32KB would be required to test the QL of 32KB. This Bulk Transfer Test can be accomplished using iperf as described in Appendix A.

The test metrics will be recorded per the stateful metrics defined in 4.2, primarily the TCP Test Pattern Execution Time (TTPET), TCP Efficiency, and Buffer Delay.

In addition to a Bulk Transfer Test, it is recommended to run the Bursty Test Pattern from appendix B at a minimum. Other tests from include: Small Web Site, Email, Citrix, etc.

The traffic is bi-directional - the same queue size is assumed for both directions.

## 6.3. Shaper tests

The intent of the shaper tests is to verify the shaper performance parameters of shape rate (SR) and shape burst size (SBS). The tests will verify that the device can handle the CIR rate with CBS and smooth the traffic bursts to the shaper rate.

Since device queues / buffers are generally an egress function, this test framework will discuss testing at the egress (although the technique can be applied to ingress and internal queues).

A network device’s traffic shaper will generally either shape to an average rate or provide settings similar to a policer to set the CIR and CBS.  In the context of a shaper, the CBS indicates the size of the burst that the shaper can accept within the shaping time interval.

The shaping time interval depends upon whether the average method os CIR/CBS method is supported by the network device.  If only the average method is supported, then the shaping time interval (period at which bursts will be shaped) must be determined through manufacturer product specifications.

For shapers that utilize the CIR/CBS method, the shaper time interval is the same as Tc for the policer which is indicated in section 6.1.

(TC comment: We need to be able to measure FD over a shaper.  That should be the ms of queue depth.)

### 6.3.1. Testing Shaper with Stateless Traffic

A traffic shaper is memory based like a queue, but with the added intelligence of an active shaping element. The same concepts from section 6.2 (Queue testing) can be applied to testing network device shaper.

The device’s traffic shaping function should be configured to the desired SR and SBS (for devices supporting this parameter) and then stateless traffic should be transmitted to test the SBS.

The same example from section 6.1 is used with SBS of 64KB and CIR of 100 Mbps; both ingress and egress ports are GigE. The Tc equates to 5.12 msec and the 64KB burst should be transmitted into the ingress port at full GigE rate, then wait for 5.12 msec for the next burst, etc.

While the ingress traffic will burst up to GigE link speed for the duration of the SBS burst, the egress traffic should be smoothed or averaged to the CIR rate on the egress port.

In addition to the egress metrics to be measured per section 4.1, the stateless shaper test shall record:

- Average shaper rate on the egress port

- Variation (min, max) around the shaper rate

## 6.3.2. Testing Shaper with Stateful Traffic

To provide a more realistic benchmark and to test queues in layer 4 devices such as firewalls, stateful traffic testing is also recommended for the shaper tests.  Stateful traffic tests will also utilize the Network Delay Emulator (NDE) from the network set-up configuration in section 2.

The BDP of the TCP test traffic must be calculated as described in section 6.2.2. To properly stress network buffers and the traffic shaping function, the cumulative TCP window should exceed the BDP which will stress the shaper.  BDP factors of 1.1 to 1.5 are recommended, but the values are the discretion of the tester and should be documented.

By cumulative TCP window, this equates to:

TCP window size* for each connection x number of connections

* TCP window size is used per RFC 6349 and is the minimum of the TCP WIN and the Send Socket Buffer (SSB)

Example, if the BDP is equal to 256 Kbytes and a connection size of 64Kbytes is used for each connection, then it would require four (4) connections to fill the BDP and 5-6 connections (over subscribe the BDP) to stress test the traffic shaping function.

Two types of tests are recommended: Bulk Transfer test and Bursty Test Pattern as documented in Appendix B at a minimum.  Other tests from include: Small Web Site, Email, Citrix, etc.

The test metrics will be recorded per the stateful metrics defined in 4.2, primarily the TCP Test Pattern Execution Time (TTPET), TCP Efficiency, and Buffer Delay.

The traffic is bi-directional involving multiple egress ports.

In addition to the egress metrics to be measured per section 4.2, the stateful shaper test shall record:

- Average shaper rate on each egress port

- Variation (min, max) around the shaper rate

## 6.4. Congestion Management tests

The intent of the congestion management tests is to benchmark the performance of various active queue management (AQM) discard techniques such as RED, WRED, etc.  AQM techniques vary, but the basic principal is to discard traffic before the queue overflows (FIFO).  This discard in effect sends congestion notification warning to protocols such as TCP, which causes TCP to back-off and ideally improves aggregate throughput by preventing global TCP session loss (tail drop).

The key parameter for AQM techniques is the discard threshold of the queue. (RK comment: The discard is also probabilistic http://en.wikipedia.org/wiki/Random_early_detection).  In some network devices, this discard threshold is discretely configurable (i.e. percent of queue depth) and in others the discard threshold is intrinsic to the AQM technique itself.

As such AQM benchmark testing may involve a certain level of characterization experiments in which the burst size transmitted may increase as a portion of the queue depth.

### 6.4.1. Testing Congestion Management with Stateless Traffic

If the queue discard threshold is discreetly configurable, then the stateless burst techniques described in sections 6.2.1 (queuing tests) can be applied directly to the AQM tests.  In other words, the queue will be over-subscribed and burst transmitted into the device within the Ti interval as defined in 6.2.1

For AQM techniques where the discard threshold is not discreetly configurable, then a stair case ramp is recommended to characterize and compare the AQM technique between devices.  For example if the QL = 32KB, then it would be reasonable to test with burst sizes in increments of 25% to include 8KB, 16KB, 32KB and record the metrics per section 4.2.  (RK comment: We should send a burst and examine if there are discontinuous drops - in the case of tail drop, the drops will be continuous)

### 6.4.2. Testing Congestion Management with Stateful Traffic

Similar to the Queue tests (section 6.2) and Shaper tests (section 6.3), stateful traffic tests will utilize the Network Delay Emulator (NDE) to add RTT.  The RTT should be configured such that BDP would equal at least 64KB.

The key metric to be measured for the stateful tests is the TCP Test Pattern Execution Time (TTPET).  AQM is intended to improve TCP performance by preventing tail-drop and it is the TTPET that provides the appropriate metric to compare the AQM techniques between vendors.

An example is as follows: transmit n TCP flows using the AQM Test Pattern (reference Appendix B) and measure the TTPET with and without AQM enabled.  The number of flows should be configured to exceed the BDP with recommended oversubscription within the 1.1 - 1.5 range.

The test metrics will be recorded per the stateful metrics defined in 4.2, primarily the TCP Test Pattern Execution Time (TTPET), TCP Efficiency, and Buffer Delay.

(TCP miscellaneous comments:

You don’t talk about impacts of RED on independent flows on testing congestion management Do certain flows get impacted more than others.

There is no discussion of SPQ versus WFQ, or any mention of QOS measurements.  We also need To make recommendations on QOS parameters / variables for acting on.

There was no discussion of UDP

There was no discussion calculating window size

)

## Appendix A: Open Source Tools for Traffic Management Testing

This traffic management framework specified that both stateless and stateful traffic testing be conducted.  Two (2) open source tools that can be used are iperf and Flowgrind to accomplish many of the tests proposed in this framework.

Iperf can generate UDP or TCP based traffic; a client and server must both run the iperf software in the same traffic mode.  The server is set up to listen and then the test traffic is controlled from the client.  Both uni-directional and bi-directional concurrent testing are supported.

The UDP mode can be used for the stateless traffic testing.  The target bandwidth, frame size, UDP port, and test duration can be controlled.  A report of bytes transmitted, frames lost, and delay variation are provided by the iperf receiver.

The TCP mode can be used for stateful traffic testing to test bulk transfer traffic.  The TCP Window size (which is actually the SSB), the number of connections, the frame size, TCP port and the test duration can be controlled.  A report of bytes transmitted and throughput achieved are provided by the iperf sender.

Flowgrind is a distributed network performance measurement tool. Using the flowgrind controller, tests can be setup between hosts running flowgrind.  For the purposes of this traffic management testing framework, the key benefit of Flowgrind is that it can emulate non-bulk transfer applications such as HTTP, Email, etc. This is due to fact that Flowgrind supports the concept of request and response behavior while iperf does not.

Traffic generation options include the request size, response size, inter-request gap, and response time gap.  Additionally, various distribution types are supported including constant, normal, exponential, pareto, etc.  These powerful traffic generation parameters facilitate the modeling of complex application test patterns at the TCP layer which are discussed in Appendix B.

Since these tools are software based, the host hardware must be qualified to be capable of generating the target traffic loads without frame loss and within the frame delay variation threshold.

## Appendix B: Stateful TCP Test Patterns

This framework does not specify a fixed set of TCP test patterns, but proposes two (2) techniques to develop standard TCP test patterns for traffic management benchmarking and provides examples of the following test patterns:

- Bulk: generate concurrent TCP connections transmit an aggregate number of in-flight data bytes (i.e. could be the BDP).  Guidelines from RFC 6349 are used to create this traffic model.

- Bursty: generate precise burst pattern within a single or multiple TCP sessions.  The idea is for TCP to establish equilibrium on a connection(s) and then to burst application bytes at a defined burst size.

- AQM: generate various burst sizes within an TCP session, spacing the bursts apart such that size of the burst size achieved (BSA) can be easily determined.  In a sense, this could be considered a TCP stair case or ramp test.

- Small Web Site: mimic the request and response (chatty) and bulk transfer (page download) behavior of a less complex web site. This example uses the modeling technique with Flowgrind to generate this TCP test pattern.

- Cirix: mimic very chatty behavior of Citrix. This example uses the packet capture technique to model the behavior and discusses the requirements for test tools to playback the packet capture statefully.

TBD: Detailed definitions for each of the test patterns listed above.

From these examples, users can extrapolate others that may be more suitable to their intended test needs.

## 7. Security Considerations

## 8. IANA Considerations

## 9. Conclusions

## 10. References

### 10.1. Normative References

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

[2]   Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

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

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

## 10.2. Informative References

## 11. Acknowledgments

## 12. First Appendix

## Authors' Addresses

Barry Constantine

JDSU, Test and Measurement Division

Germantown, MD 20876-7100, USA

Phone: +1 240 404 2227

Email: barry.constantine@jdsu.com

Timothy Copley

Level 3 Communications

14605 S 50th Street

Phoenix, AZ 85044

Email: Timothy.copley@level3.com

Ram Krishnan

Brocade Communications

San Jose, 95134, USA

Phone: +001-408-406-7890

Email: ramk@brocade.com