Internet Engineering Task Force (IETF)
Request for Comments: 7640
Category: Informational
ISSN: 2070-1721

B. Constantine
JDSU
R. Krishnan
Dell Inc.

September 2015

Traffic Management Benchmarking

Abstract

This framework describes a practical methodology for benchmarking the traffic management capabilities of networking devices (i.e., policing, shaping, etc.). The goals are 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.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7640.

Table of Contents

1. Introduction

Traffic management (i.e., policing, shaping, etc.) is an increasingly important component when implementing network Quality of Service (QoS).

There is currently no framework to benchmark these features, although some standards address specific areas as described in Section 1.1.

This document provides a framework to conduct repeatable traffic management benchmarks for devices and systems in a lab environment.

Specifically, this framework defines the methods to characterize the capacity of the following traffic management features in network devices: classification, policing, queuing/scheduling, and traffic shaping.

This benchmarking framework can also be used as a test procedure to assist in the tuning of traffic management parameters before service activation. In addition to Layer 2/3 (Ethernet/IP) benchmarking, Layer 4 (TCP) test patterns are proposed by this document in order to more realistically benchmark end-user traffic.

1.1. Traffic Management Overview

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

- Traffic classification: identifies traffic according to various configuration rules (for example, IEEE 802.1Q Virtual LAN (VLAN), Differentiated Services Code Point (DSCP)) and marks this traffic internally to the network device. Multiple external priorities (DSCP, 802.1p, etc.) can map to the same priority in the device.

- Traffic policing: limits the rate of traffic that enters a network device according to the traffic classification. If the traffic exceeds the provisioned limits, the traffic is either dropped or remarked and forwarded onto the next network device.

- Traffic scheduling: provides traffic classification within the network device by directing packets to various types of queues and applies a dispatching algorithm to assign the forwarding sequence of packets.

- Traffic shaping: controls traffic by actively buffering and smoothing the output rate in an attempt to adapt bursty traffic to the configured limits.

- Active Queue Management (AQM): involves monitoring the status of internal queues and proactively dropping (or remarking) packets, which causes hosts using congestion-aware protocols to "back off" and in turn alleviate queue congestion [RFC7567]. On the other hand, classic traffic management techniques reactively drop (or remark) packets based on queue-full conditions. The benchmarking scenarios for AQM are different and are outside the scope of this testing framework.

Even though AQM is outside the scope of this framework, it should be noted that the TCP metrics and TCP test patterns (defined in Sections 4.2 and 5.2, respectively) could be useful to test new AQM algorithms (targeted to alleviate "bufferbloat"). Examples of these algorithms include Controlled Delay [CoDel] and Proportional Integral controller Enhanced [PIE].

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 it provides context for this test framework.

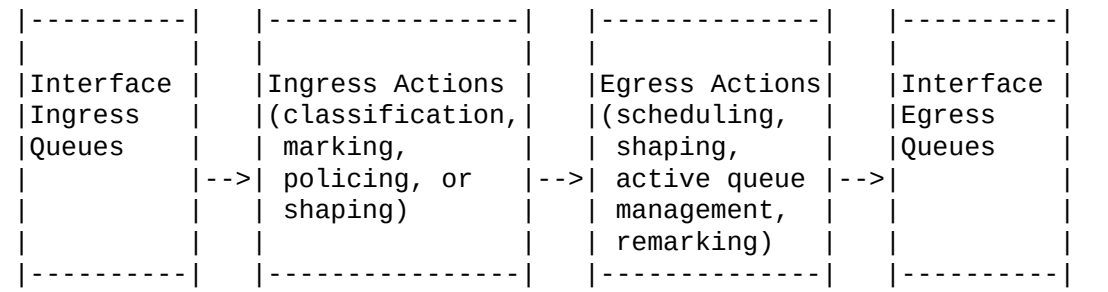

Figure 1: Generic Traffic Management Capabilities of a Network Device

Ingress actions such as classification are defined in [RFC4689] and include IP addresses, port numbers, and DSCP. In terms of marking, [RFC2697] and [RFC2698] define a Single Rate Three Color Marker and a Two Rate Three Color Marker, respectively.

The Metro Ethernet Forum (MEF) specifies policing and shaping in terms of ingress and egress subscriber/provider conditioning functions as described in MEF 12.2 [MEF-12.2], as well as ingress and bandwidth profile attributes as described in MEF 10.3 [MEF-10.3] and MEF 26.1 [MEF-26.1].

1.2. Lab Configuration and Testing Overview

The following diagram shows the lab setup for the traffic management tests:

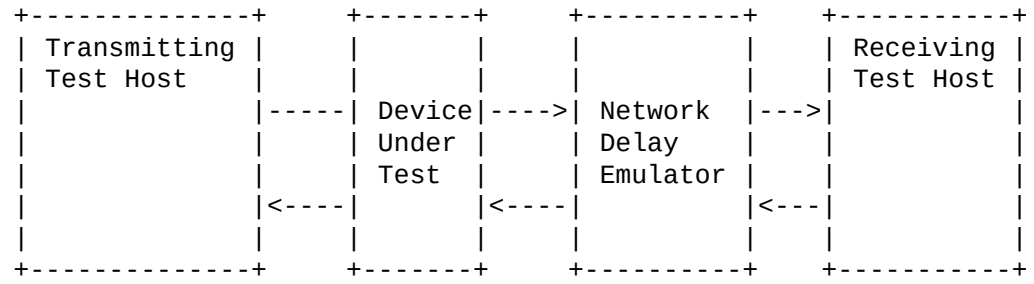

Figure 2: Lab Setup for Traffic Management Tests

As shown in the test diagram, the framework supports unidirectional and bidirectional traffic management tests (where the transmitting and receiving roles would be reversed on the return path).

This testing framework describes the tests and metrics for each of the following traffic management functions:

- Classification

- Policing

- Queuing/scheduling

- Shaping

The tests are divided into individual and rated capacity tests. The individual tests are intended to benchmark the traffic management functions according to the metrics defined in Section 4. The capacity tests verify traffic management functions under the load of many simultaneous individual tests and their flows.

This involves concurrent testing of multiple interfaces with the specific traffic management function enabled, and increasing the load to the capacity limit of each interface.

For example, a device is specified to be capable of shaping on all of its egress ports. The individual test would first be conducted to benchmark the specified shaping function against the metrics defined in Section 4. Then, the capacity test would be executed to test the shaping function concurrently on all interfaces and with maximum traffic load.

The Network Delay Emulator (NDE) is required for TCP stateful tests in order to allow TCP to utilize a TCP window of significant size in its control loop.

Note also that the NDE SHOULD be passive in nature (e.g., 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 where 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. In other words, the Device Under Test (DUT) would be removed and the NDE performance benchmarked independently.

Note that the NDE SHOULD be used only as emulated delay. Most NDEs allow for per-flow delay actions, emulating QoS prioritization. For this framework, the NDE's sole purpose is simply to add delay to all packets (emulate network latency). So, to benchmark the performance of the NDE, the maximum offered load should be tested against the following frame sizes: 128, 256, 512, 768, 1024, 1500, and 9600 bytes. The delay accuracy at each of these packet sizes can then be used to calibrate the range of expected Bandwidth-Delay Product (BDP) for the TCP stateful tests.

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 [RFC2119].

The following acronyms are used:

AQM: Active Queue Management

BB: Bottleneck Bandwidth

BDP: Bandwidth-Delay Product

BSA: Burst Size Achieved

CBS: Committed Burst Size

CIR: Committed Information Rate

DUT: Device Under Test

EBS: Excess Burst Size

EIR: Excess Information Rate

NDE: Network Delay Emulator

QL: Queue Length

QoS: Quality of Service

RTT: Round-Trip Time

SBB: Shaper Burst Bytes

SBI: Shaper Burst Interval

SP: Strict Priority

SR: Shaper Rate

SSB: Send Socket Buffer

SUT: System Under Test

Ti: Transmission Interval

TTP: TCP Test Pattern

TTPET: TCP Test Pattern Execution Time

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
- General Layer 4-7 appliances (Proxies, WAN Accelerators, etc.)

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

The primary goal is to assess the maximum forwarding performance deemed to be within the provisioned traffic limits that a network device can sustain without dropping or impairing packets, and without compromising the accuracy of multiple instances of traffic management functions. This is the benchmark for comparison between devices.

Within this framework, the metrics are defined for each traffic management test but do not include pass/fail criteria, which are not within the charter of the BMWG. This framework provides the test methods and metrics to conduct repeatable testing, which will provide the means to compare measured performance between DUTs.

As mentioned in Section 1.2, these methods describe the individual tests and metrics for several management functions. It is also within scope that this framework will benchmark each function in terms of overall rated capacity. This involves concurrent testing of multiple interfaces with the specific traffic management function enabled, up to the capacity limit of each interface.

It is not within the scope of this framework to specify the procedure for testing multiple configurations of traffic management functions concurrently. The multitudes of possible combinations are almost unbounded, and the ability to identify functional "break points" would be almost impossible.

However, Section 6.4 provides suggestions for some profiles of concurrent functions that would be useful to benchmark. The key requirement for any concurrent test function is that tests MUST produce reliable and repeatable results.

Also, it is not within scope to perform conformance testing. Tests defined in this framework benchmark the traffic management functions according to the metrics defined in Section 4 and do not address any conformance to standards related to traffic management.

The current specifications don’t specify exact behavior or implementation, and the specifications that do exist (cited in Section 1.1) allow implementations to vary with regard to short-term rate accuracy and other factors. This is a primary driver for this framework: to provide an objective means to compare vendor traffic management functions.

Another goal is to devise methods that utilize flows with congestion-aware transport (TCP) as part of the traffic load and still produce repeatable results in the isolated test environment. This framework will 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 queuing/scheduling and traffic shaping. In cases where the network device is stateful in nature (i.e., firewall, etc.), stateful test pattern traffic is important to test, along with stateless UDP traffic in specific test scenarios (i.e., applications using TCP transport and UDP VoIP, etc.).

As mentioned earlier in this document, repeatability of test results is critical, especially considering the nature of stateful TCP traffic. To this end, the stateful tests will use TCP test patterns to emulate applications. This framework also provides guidelines for application modeling and open source tools to achieve the repeatable stimulus. Finally, TCP metrics from [RFC6349] MUST be measured for each stateful test and provide the means to compare each repeated test.

Even though this framework targets the testing of TCP applications (i.e., web, email, database, etc.), it could also be applied to the Stream Control Transmission Protocol (SCTP) in terms of test patterns. WebRTC, Signaling System 7 (SS7) signaling, and 3GPP are SCTP-based applications that could be modeled with this framework to benchmark SCTP's effect on traffic management performance.

Note that at the time of this writing, this framework does not address tcpcrypt (encrypted TCP) test patterns, although the metrics defined in Section 4.2 can still be used because the metrics are based on TCP retransmission and RTT measurements (versus any of the payload). Thus, if tcpcrypt becomes popular, it would be natural for benchmarkers to consider encrypted TCP patterns and include them in test cases.

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 TCP-layer metrics used for the stateful traffic testing.

4.1. Metrics for Stateless Traffic Tests

Stateless traffic measurements require that a sequence number and timestamp be inserted into the payload for lost-packet analysis. Delay analysis may be achieved by insertion of timestamps directly into the packets or timestamps stored elsewhere (packet captures). This framework does not specify the packet format to carry sequence number or timing information.

However, [RFC4737] and [RFC4689] provide recommendations for sequence tracking, along with definitions of in-sequence and out-of-order packets.

The following metrics MUST be measured 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 Excess Burst Size (EBS) of a policer or the queue/buffer size configured in the DUT. The BSA metric is a measure of the actual burst size received at the egress port of the DUT with no lost packets. For example, the configured CBS of a DUT is 64 KB, and after the burst test, only a 63 KB burst can be achieved without packet loss. Then, 63 KB is the BSA. Also, the average Packet Delay Variation (PDV) (see below) as experienced by the packets sent at the BSA burst size should be recorded. This metric SHALL be reported in units of bytes, KB, or MB.

- Lost Packets (LP): For all traffic management tests, the tester will transmit the test packets into the DUT ingress port, and the number of packets received at the egress port will be measured. The difference between packets transmitted into the ingress port and received at the egress port is the number of lost packets as measured at the egress port. These packets must have unique identifiers such that only the test packets are measured. For cases where multiple flows are transmitted from the ingress port to the egress port (e.g., IP conversations), each flow must have sequence numbers within the stream of test packets.

[RFC6703] and [RFC2680] describe the need to establish the time threshold to wait before a packet is declared as lost. This threshold MUST be reported, with the results reported as an integer number that cannot be negative.

- Out-of-Sequence (OOS): In addition to the LP metric, the test packets must be monitored for sequence. [RFC4689] defines the general function of sequence tracking, as well as definitions for in-sequence and out-of-order packets. Out-of-order packets will be counted per [RFC4737]. This metric SHALL be reported as an integer number that cannot be negative.

- Packet Delay (PD): The PD metric is the difference between the timestamp of the received egress port packets and the packets transmitted into the ingress port, as specified in [RFC1242]. The transmitting host and receiving host time must be in time sync (achieved by using NTP, GPS, etc.). This metric SHALL be reported as a real number of seconds, where a negative measurement usually indicates a time synchronization problem between test devices.

- Packet Delay Variation (PDV): The PDV metric is the variation between the timestamp of the received egress port packets, as specified in [RFC5481]. Note that per [RFC5481], this PDV is the variation of one-way delay across many packets in the traffic flow. Per the measurement formula in [RFC5481], select the high percentile of 99%, and units of measure will be a real number of seconds (a negative value is not possible for the PDV and would indicate a measurement error).

- Shaper Rate (SR): The SR represents the average DUT output rate (bps) over the test interval. The SR is only applicable to the traffic-shaping tests.

- Shaper Burst Bytes (SBB): A traffic shaper will emit packets in "trains" of different sizes; these frames are emitted "back-to-back" with respect to the mandatory interframe gap. This metric characterizes the method by which the shaper emits traffic. Some shapers transmit larger bursts per interval, and a burst of one packet would apply to the less common case of a shaper sending a constant-bitrate stream of single packets. This metric SHALL be reported in units of bytes, KB, or MB. The SBB metric is only applicable to the traffic-shaping tests.

- Shaper Burst Interval (SBI): The SBI is the time between bursts emitted by the shaper and is measured at the DUT egress port. This metric SHALL be reported as a real number of seconds. The SBI is only applicable to the traffic-shaping tests.

4.2. Metrics for Stateful Traffic Tests

The stateful metrics will be based on [RFC6349] TCP metrics and MUST include:

- TCP Test Pattern Execution Time (TTPET): [RFC6349] 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 include bulk transfer and interactive applications. The interactive patterns include instances such as HTTP business applications and database applications. The TTPET will be the measure of the time for a single execution of a TCP Test Pattern (TTP). Average, minimum, and maximum times will be measured or calculated and expressed as a real number of seconds.

An example would be an interactive HTTP TTP session that should take 5 seconds on a GigE network with 0.5-millisecond latency. During ten (10) executions of this TTP, the TTPET results might be an average of 6.5 seconds, a minimum of 5.0 seconds, and a maximum of 7.9 seconds.

- TCP Efficiency: After the execution of the TTP, TCP Efficiency represents the percentage of bytes that were not retransmitted.

$$\text{TCP Efficiency \%} = \frac{\text{Transmitted Bytes} - \text{Retransmitted Bytes}}{\text{Transmitted Bytes}} \times 100$$

"Transmitted Bytes" is the total number of TCP bytes to be transmitted, including the original bytes and the retransmitted bytes. To avoid any misinterpretation that a reordered packet is a retransmitted packet (as may be the case with packet decode interpretation), these retransmitted bytes should be recorded from the perspective of the sender's TCP/IP stack.

- Buffer Delay: Buffer Delay represents the increase in RTT during a TCP test versus the baseline DUT RTT (non-congested, inherent latency). RTT and the technique to measure RTT (average versus baseline) are defined in [RFC6349]. Referencing [RFC6349], the average RTT is derived from the total of all measured RTTs during the actual test sampled at every second divided by the test duration in seconds.

$$\text{Average RTT during transfer} = \frac{\text{Total RTTs during transfer}}{\text{Transfer duration in seconds}}$$

$$\text{Buffer Delay \%} = \frac{\text{Average RTT during transfer} - \text{Baseline RTT}}{\text{Baseline RTT}} \times 100$$

Note that even though this was not explicitly stated in [RFC6349], retransmitted packets should not be used in RTT measurements.

Also, the test results should record the average RTT in milliseconds across the entire test duration, as well as the number of samples.

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 device MUST be capable of generating traffic at up to the link speed of the DUT. The test device must be calibrated to verify that it will not drop any packets. The test device's inherent PD and PDV must also be calibrated and subtracted from the PD and PDV metrics. The test device must support the encapsulation to be tested, e.g., IEEE 802.1Q VLAN, IEEE 802.1ad Q-in-Q, Multiprotocol Label Switching (MPLS). Also, the test device must allow control of the classification techniques defined in [RFC4689] (e.g., IP address, DSCP, classification of Type of Service).

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 (e.g., 1 GigE, 10 GigE). Careful calibration of any test environment using iperf is important. At higher link speeds, using hardware-based packet test equipment is recommended.

5.1.1. Burst Hunt with Stateless Traffic

A central theme for the traffic management tests is to benchmark the specified burst parameter of a traffic management function, since burst parameters listed in Service Level Agreements (SLAs) are specified in bytes. For testing efficiency, including a burst hunt feature is recommended, as this feature automates the manual process of determining the maximum burst size that can be supported by a traffic management function.

The burst hunt algorithm should start at the target burst size (maximum burst size supported by the traffic management function) and will send single bursts until it can determine the largest burst that can pass without loss. If the target burst size passes, then the test is complete. The "hunt" aspect occurs when the target burst size is not achieved; the algorithm will drop down to a configured minimum burst size and incrementally increase the burst until the maximum burst supported by the DUT is discovered. The recommended granularity of the incremental burst size increase is 1 KB.

For a policer function, if the burst size passes, the burst should be increased by increments of 1 KB to verify that the policer is truly configured properly (or enabled at all).

5.2. Stateful Test Pattern Generation

The TCP test host will have many of the same attributes as the TCP test host defined in [RFC6349]. The TCP test device 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 (the NDE function as shown in the lab setup diagram in Section 1.2) must be used in order to provide a meaningful BDP. As discussed in Section 1.2, the target traffic rate and configured RTT MUST be verified independently, using just the NDE for all stateful tests (to ensure that the NDE can add delay without inducing any packet loss).

The TCP test host MUST be capable of generating and receiving 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 500 Mbps may require high-performance server hardware or dedicated hardware-based test tools.

The TCP test host MUST allow the adjustment of both Send and Receive Socket Buffer sizes. The Socket Buffers must be large enough to fill the BDP for bulk transfer of 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 (e.g., TCP New Reno, TCP options supported). Additionally, the test results SHALL provide specific congestion control algorithm details, as per [RFC3148].

While [RFC6349] 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, and business applications. This interactive traffic is bidirectional and can be chatty, meaning many turns in traffic communication during the course of a transaction (versus the relatively unidirectional flow of bulk transfer applications).

The test device must not only support bulk TCP transfer application traffic but MUST also support chatty traffic. A valid stress test SHOULD include both traffic types. 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 "netperf" open source tool provides the ability to control client and server request/response behavior. The netperf-wrapper tool is a Python script that runs multiple simultaneous netperf instances and aggregates the results. Appendix A provides an overview of netperf/netperf-wrapper, as well as iperf. As with any software-based tool, the performance must be qualified to the link speed to be tested. Hardware-based test equipment should be considered for reliable results at higher link speeds (e.g., 1 GigE, 10 GigE).

5.2.1. TCP Test Pattern Definitions

As mentioned in the goals of this framework, techniques are defined to specify TCP traffic test patterns to benchmark traffic management technique(s) and produce repeatable results. Some network devices, such as firewalls, will not process stateless test traffic; this is another reason why stateful TCP test traffic must be used.

An application could be fully emulated up to Layer 7; however, this framework proposes that stateful TCP test patterns be used in order to provide granular and repeatable control for the benchmarks. The following diagram illustrates a simple web-browsing application (HTTP).

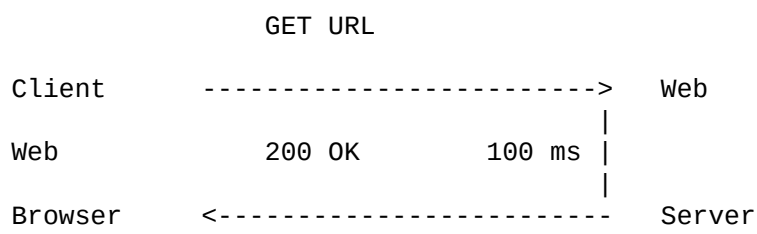

Figure 3: Simple Flow Diagram for a Web Application

In this example, the Client Web Browser (client) requests a URL, and then the Web Server delivers the web page content to the client (after a server delay of 100 milliseconds). This asynchronous "request/response" behavior is intrinsic to most TCP-based applications, such as email (SMTP), file transfers (FTP and Server Message Block (SMB)), database (SQL), web applications (SOAP), and Representational State Transfer (REST). The impact on the network elements is due to the multitudes of clients and the variety of bursty traffic, which stress traffic management functions. The actual emulation of the specific application protocols is not required, and TCP test patterns can be defined to mimic the application network traffic flows and produce repeatable results.

Application modeling techniques have been proposed in [3GPP2-C_R1002-A], which provides examples to model the behavior of HTTP, FTP, and Wireless Application Protocol (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 model definition formats described in [3GPP2-C_R1002-A] are the basis for the guidelines provided in Appendix B and are also similar to formats used by network modeling tools. Packet captures can also be used to characterize application traffic and specify some of the test patterns listed in Appendix B.

This framework does not specify a fixed set of TCP test patterns but does provide test cases that SHOULD be performed; see Appendix B. Some of these examples reflect those specified in [CA-Benchmark], which suggests traffic mixes for a variety of representative application profiles. Other examples are simply well-known application traffic types such as HTTP.

6. Traffic Benchmarking Methodology

The traffic benchmarking methodology uses the test setup from Section 1.2 and metrics defined in Section 4.

Each test SHOULD compare the network device's internal statistics (available via command line management interface, SNMP, etc.) to the measured metrics defined in Section 4. This evaluates the accuracy of the internal traffic management counters under individual test conditions and capacity test conditions as defined in Sections 4.1 and 4.2. This comparison is not intended to compare real-time statistics, but rather the cumulative statistics reported after the test has completed and device counters have updated (it is common for device counters to update after an interval of 10 seconds or more).

From a device configuration standpoint, scheduling and shaping functionality can be applied to logical ports (e.g., Link Aggregation (LAG)). This would result in the same scheduling and shaping configuration applied to all of the member physical ports. The focus of this document is only on tests at a physical-port level.

The following sections provide the objective, procedure, metrics, and reporting format for each test. For all test steps, the following global parameters must be specified:

Test Runs (Tr):
: The number of times the test needs to be run to ensure accurate and repeatable results. The recommended value is a minimum of 10.

Test Duration (Td):
: The duration of a test iteration, expressed in seconds. The recommended minimum value is 60 seconds.

The variability in the test results MUST be measured between test runs, and if the variation is characterized as a significant portion of the measured values, the next step may be to revise the methods to achieve better consistency.

6.1. Policing Tests

A policer is defined as the entity performing the policy function. The intent of the policing tests is to verify the policer performance (i.e., CIR/CBS and EIR/EBS parameters). The tests will verify that the network device can handle the CIR with CBS and the EIR with EBS, and will use back-to-back packet-testing concepts as described in [RFC2544] (but adapted to burst size algorithms and terminology). Also, [MEF-14], [MEF-19], and [MEF-37] provide some bases for

specific components of this test. The burst hunt algorithm defined in Section 5.1.1 can also be used to automate the measurement of the CBS value.

The tests are divided into two (2) sections: individual policer tests and then full-capacity policing tests. It is important to benchmark the basic functionality of the individual policer and then proceed into the fully rated capacity of the device. This capacity may include the number of policing policies per device and the number of policers simultaneously active across all ports.

6.1.1. Policer Individual Tests

Objective:

Test a policer as defined by [RFC4115] or [MEF-10.3], depending upon the equipment's specification. 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 remark or drop excess packets, and pass traffic at the specified CBS/EBS values.

Test Summary:

Policing tests should use stateless traffic. Stateful TCP test traffic will generally be adversely affected by a policer in the absence of traffic shaping. So, while TCP traffic could be used, it is more accurate to benchmark a policer with stateless traffic.

As an example of a policer as defined by [RFC4115], consider a CBS/EBS of 64 KB and CIR/EIR of 100 Mbps on a 1 GigE physical link (in color-blind mode). A stateless traffic burst of 64 KB would be sent into the policer at the GigE rate. This equates to an approximately 0.512-millisecond burst time (64 KB at 1 GigE). The traffic generator must space these bursts to ensure that the aggregate throughput does not exceed the CIR. The Ti between the bursts would equal CBS * 8 / CIR = 5.12 milliseconds in this example.

Test Metrics:

The metrics defined in Section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

Procedure:

1. Configure the DUT policing parameters for the desired CIR/EIR and CBS/EBS values to be tested.

2. Configure the tester to generate a stateless traffic burst equal to CBS and an interval equal to Ti (CBS in bits/CIR).

3. Compliant Traffic Test: Generate bursts of CBS + EBS traffic into the policer ingress port, and measure the metrics defined in Section 4.1 (BSA, LP, OOS, PD, and PDV) at the egress port and across the entire Td (default 60-second duration).

4. Excess Traffic Test: Generate bursts of greater than CBS + EBS bytes into the policer ingress port, and verify that the policer only allowed the BSA bytes to exit the egress. The excess burst MUST be recorded; the recommended value is 1000 bytes. Additional tests beyond the simple color-blind example might include color-aware mode, configurations where EIR is greater than CIR, etc.

Reporting Format:
The policer individual report MUST contain all results for each CIR/EIR/CBS/EBS test run. A recommended format is as follows:

****

Test Configuration Summary: Tr, Td

DUT Configuration Summary: CIR, EIR, CBS, EBS

The results table should contain entries for each test run, as follows (Test #1 to Test #Tr):

- Compliant Traffic Test: BSA, LP, OOS, PD, and PDV

- Excess Traffic Test: BSA

****

6.1.2. Policer Capacity Tests

Objective:
The intent of the capacity tests is to verify the policer performance in a scaled environment with multiple ingress customer policers on multiple physical ports. This test will benchmark the maximum number of active policers as specified by the device manufacturer.

Test Summary:
The specified policing function capacity is generally expressed in terms of the number of policers active on each individual physical port as well as the number of unique policer rates that are utilized. For all of the capacity tests, the benchmarking test

procedure and reporting format described in Section 6.1.1 for a single policer MUST be applied to each of the physical-port policers.

For example, a Layer 2 switching device may specify that each of the 32 physical ports can be policed using a pool of policing service policies. The device may carry a single customer's traffic on each physical port, and a single policer is instantiated per physical port. Another possibility is that a single physical port may carry multiple customers, in which case many customer flows would be policed concurrently on an individual physical port (separate policers per customer on an individual port).

Test Metrics:
The metrics defined in Section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

The following sections provide the specific test scenarios, procedures, and reporting formats for each policer capacity test.

6.1.2.1. Maximum Policers on Single Physical Port

Test Summary:
The first policer capacity test will benchmark a single physical port, with maximum policers on that physical port.

Assume multiple categories of ingress policers at rates r1, r2, ..., rn. There are multiple customers on a single physical port. Each customer could be represented by a single-tagged VLAN, a double-tagged VLAN, a Virtual Private LAN Service (VPLS) instance, etc. Each customer is mapped to a different policer. Each of the policers can be of rates r1, r2, ..., rn.

An example configuration would be

- Y1 customers, policer rate r1

- Y2 customers, policer rate r2

- Y3 customers, policer rate r3

...

- Yn customers, policer rate rn

Some bandwidth on the physical port is dedicated for other traffic (i.e., other than customer traffic); this includes network control protocol traffic. There is a separate policer for the other traffic. Typical deployments have three categories of policers; there may be some deployments with more or less than three categories of ingress policers.

Procedure:

1. Configure the DUT policing parameters for the desired CIR/EIR and CBS/EBS values for each policer rate (r1-rn) to be tested.

2. Configure the tester to generate a stateless traffic burst equal to CBS and an interval equal to Ti (CBS in bits/CIR) for each customer stream (Y1-Yn). The encapsulation for each customer must also be configured according to the service tested (VLAN, VPLS, IP mapping, etc.).

3. Compliant Traffic Test: Generate bursts of CBS + EBS traffic into the policer ingress port for each customer traffic stream, and measure the metrics defined in Section 4.1 (BSA, LP, OOS, PD, and PDV) at the egress port for each stream and across the entire Td (default 30-second duration).

4. Excess Traffic Test: Generate bursts of greater than CBS + EBS bytes into the policer ingress port for each customer traffic stream, and verify that the policer only allowed the BSA bytes to exit the egress for each stream. The excess burst MUST be recorded; the recommended value is 1000 bytes.

Reporting Format:

The policer individual report MUST contain all results for each CIR/EIR/CBS/EBS test run, per customer traffic stream. A recommended format is as follows:

****

Test Configuration Summary: Tr, Td

Customer Traffic Stream Encapsulation: Map each stream to VLAN, VPLS, IP address

DUT Configuration Summary per Customer Traffic Stream: CIR, EIR, CBS, EBS

The results table should contain entries for each test run, as follows (Test #1 to Test #Tr):

- Customer Stream Y1-Yn (see note) Compliant Traffic Test: BSA, LP, OOS, PD, and PDV

- Customer Stream Y1-Yn (see note) Excess Traffic Test: BSA

****

Note: For each test run, there will be two (2) rows for each customer stream: the Compliant Traffic Test result and the Excess Traffic Test result.

6.1.2.2. Single Policer on All Physical Ports

Test Summary:
The second policer capacity test involves a single policer function per physical port with all physical ports active. In this test, there is a single policer per physical port. The policer can have one of the rates r1, r2, ..., rn. All of the physical ports in the networking device are active.

Procedure:
The procedure for this test is identical to the procedure listed in Section 6.1.1. The configured parameters must be reported per port, and the test report must include results per measured egress port.

6.1.2.3. Maximum Policers on All Physical Ports

The third policer capacity test is a combination of the first and second capacity tests, i.e., maximum policers active per physical port and all physical ports active.

Procedure:
The procedure for this test is identical to the procedure listed in Section 6.1.2.1. The configured parameters must be reported per port, and the test report must include per-stream results per measured egress port.

6.2. Queue/Scheduler Tests

Queues and traffic scheduling are closely related in that a queue's priority dictates the manner in which the traffic scheduler transmits packets out of the egress port.

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).

Similar to the policing tests, these tests are divided into two sections: individual queue/scheduler function tests and then full-capacity tests.

6.2.1. Queue/Scheduler Individual Tests

The various types of scheduling techniques include FIFO, Strict Priority (SP) queuing, and Weighted Fair Queuing (WFQ), along with other variations. This test framework recommends testing with a minimum of three techniques, although benchmarking other device-scheduling algorithms is left to the discretion of the tester.

6.2.1.1. Testing Queue/Scheduler with Stateless Traffic

Objective:
Verify that the configured queue and scheduling technique can handle stateless traffic bursts up to the queue depth.

Test Summary:
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 queues.

The device's network queue should be configured to the desired size in KB (i.e., Queue Length (QL)), and then stateless traffic should be transmitted to test this QL.

A queue should be able to handle repetitive bursts with the transmission gaps proportional to the Bottleneck Bandwidth (BB). The transmission gap is referred to here as the transmission interval (Ti). The Ti can be defined for the traffic bursts and is based on the QL and BB of the egress interface.

$$Ti = QL * 8 / BB$$

Note that this equation is similar to the Ti required for transmission into a policer (QL = CBS, BB = CIR). Note also that the burst hunt algorithm defined in Section 5.1.1 can also be used to automate the measurement of the queue value.

The stateless traffic burst SHALL be transmitted at the link speed and spaced within the transmission interval (Ti). The metrics defined in Section 4.1 SHALL be measured at the egress port and recorded; the primary intent is to verify the BSA and verify that no packets are dropped.

The scheduling function must also be characterized to benchmark the device's ability to schedule the queues according to the priority. An example would be two levels of priority that include SP and FIFO queuing. Under a flow load greater than the egress port speed, the higher-priority packets should be transmitted without drops (and also maintain low latency), while the lower-priority (or best-effort) queue may be dropped.

Test Metrics:

The metrics defined in Section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

Procedure:

1. Configure the DUT QL and scheduling technique parameters (FIFO, SP, etc.).

2. Configure the tester to generate a stateless traffic burst equal to QL and an interval equal to Ti (QL in bits/BB).

3. Generate bursts of QL traffic into the DUT, and measure the metrics defined in Section 4.1 (LP, OOS, PD, and PDV) at the egress port and across the entire Td (default 30-second duration).

Reporting Format:

The Queue/Scheduler Stateless Traffic individual report MUST contain all results for each QL/BB test run. A recommended format is as follows:

****

Test Configuration Summary: Tr, Td

DUT Configuration Summary: Scheduling technique (i.e., FIFO, SP, WFQ, etc.), BB, and QL

The results table should contain entries for each test run, as follows (Test #1 to Test #Tr):

- LP, OOS, PD, and PDV

****

6.2.1.2. Testing Queue/Scheduler with Stateful Traffic

Objective:
Verify that the configured queue and scheduling technique can handle stateful traffic bursts up to the queue depth.

Test Background and Summary:
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 setup configuration in Section 1.2.

The BDP of the TCP test traffic must be calibrated to the QL of the device queue. Referencing [RFC6349], the BDP is equal to:

BB * RTT / 8 (in bytes)

The NDE must be configured to an RTT value that is large 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 = 32 KB

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

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

Two types of TCP tests MUST be performed: the Bulk Transfer Test and the Micro Burst Test Pattern, as documented in Appendix B. The Bulk Transfer Test only bursts during the TCP Slow Start (or Congestion Avoidance) state, while the Micro Burst Test Pattern emulates application-layer bursting, which may occur any time during the TCP connection.

Other types of tests SHOULD include the following: simple web sites, complex web sites, business applications, email, and SMB/CIFS (Common Internet File System) file copy (all of which are also documented in Appendix B).

Test Metrics:

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

Procedure:

1. Configure the DUT QL and scheduling technique parameters (FIFO, SP, etc.).

2. Configure the test generator* with a profile of an emulated application traffic mixture.

   - The application mixture MUST be defined in terms of percentage of the total bandwidth to be tested.

   - The rate of transmission for each application within the mixture MUST also be configurable.

   * To ensure repeatable results, the test generator MUST be capable of generating precise TCP test patterns for each application specified.

3. Generate application traffic between the ingress (client side) and egress (server side) ports of the DUT, and measure the metrics (TTPET, TCP Efficiency, and Buffer Delay) per application stream and at the ingress and egress ports (across the entire Td, default 60-second duration).

A couple of items require clarification concerning application measurements: an application session may be comprised of a single TCP connection or multiple TCP connections.

If an application session utilizes a single TCP connection, the application throughput/metrics have a 1-1 relationship to the TCP connection measurements.

If an application session (e.g., an HTTP-based application) utilizes multiple TCP connections, then all of the TCP connections are aggregated in the application throughput measurement/metrics for that application.

Then, there is the case of multiple instances of an application session (i.e., multiple FTPs emulating multiple clients). In this situation, the test should measure/record each FTP application session independently, tabulating the minimum, maximum, and average for all FTP sessions.

Finally, application throughput measurements are based on Layer 4 TCP throughput and do not include bytes retransmitted. The TCP Efficiency metric MUST be measured during the test, because it provides a measure of "goodput" during each test.

Reporting Format:

The Queue/Scheduler Stateful Traffic individual report MUST contain all results for each traffic scheduler and QL/BB test run. A recommended format is as follows:

****

Test Configuration Summary: Tr, Td

DUT Configuration Summary: Scheduling technique (i.e., FIFO, SP, WFQ, etc.), BB, and QL

Application Mixture and Intensities: These are the percentages configured for each application type.

The results table should contain entries for each test run, with minimum, maximum, and average per application session, as follows (Test #1 to Test #Tr):

- Throughput (bps) and TTPET for each application session

- Bytes In and Bytes Out for each application session

- TCP Efficiency and Buffer Delay for each application session

****

6.2.2. Queue/Scheduler Capacity Tests

Objective:
The intent of these capacity tests is to benchmark queue/scheduler performance in a scaled environment with multiple queues/schedulers active on multiple egress physical ports. These tests will benchmark the maximum number of queues and schedulers as specified by the device manufacturer. Each priority in the system will map to a separate queue.

Test Metrics:
The metrics defined in Section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

The following sections provide the specific test scenarios, procedures, and reporting formats for each queue/scheduler capacity test.

6.2.2.1. Multiple Queues, Single Port Active

For the first queue/scheduler capacity test, multiple queues per port will be tested on a single physical port. In this case, all of the queues (typically eight) are active on a single physical port. Traffic from multiple ingress physical ports is directed to the same egress physical port. This will cause oversubscription on the egress physical port.

There are many types of priority schemes and combinations of priorities that are managed by the scheduler. The following sections specify the priority schemes that should be tested.

6.2.2.1.1. Strict Priority on Egress Port

Test Summary:
For this test, SP scheduling on the egress physical port should be tested, and the benchmarking methodologies specified in Sections 6.2.1.1 (stateless) and 6.2.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. For a given priority, each ingress physical port should get a fair share of the egress physical-port bandwidth.

Since this is a capacity test, the configuration and report results format (see Sections 6.2.1.1 and 6.2.1.2) MUST also include:

Configuration:

- The number of physical ingress ports active during the test
- The classification marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port

Report Results:

- For each ingress port traffic stream, the achieved throughput rate and metrics at the egress port

6.2.2.1.2. Strict Priority + WFQ on Egress Port

Test Summary:
For this test, SP and WFQ should be enabled simultaneously in the scheduler, but on a single egress port. The benchmarking methodologies specified in Sections 6.2.1.1 (stateless) and 6.2.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. Additionally, the egress port bandwidth-sharing among weighted queues should be proportional to the assigned weights. For a given priority, each ingress physical port should get a fair share of the egress physical-port bandwidth.

Since this is a capacity test, the configuration and report results format (see Sections 6.2.1.1 and 6.2.1.2) MUST also include:

Configuration:

- The number of physical ingress ports active during the test
- The classification marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port

Report Results:

- For each ingress port traffic stream, the achieved throughput rate and metrics at each queue of the egress port queue (both the SP and WFQ)

Example:

- Egress Port SP Queue: throughput and metrics for ingress streams 1-n

- Egress Port WFQ: throughput and metrics for ingress streams 1-n

6.2.2.2. Single Queue per Port, All Ports Active

Test Summary:
Traffic from multiple ingress physical ports is directed to the same egress physical port. This will cause oversubscription on the egress physical port. Also, the same amount of traffic is directed to each egress physical port.

The benchmarking methodologies specified in Sections 6.2.1.1 (stateless) and 6.2.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. Each ingress physical port should get a fair share of the egress physical-port bandwidth. Additionally, each egress physical port should receive the same amount of traffic.

Since this is a capacity test, the configuration and report results format (see Sections 6.2.1.1 and 6.2.1.2) MUST also include:

Configuration:

- The number of ingress ports active during the test

- The number of egress ports active during the test

- The classification marking (DSCP, VLAN, etc.) for each physical ingress port

- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port

Report Results:

- For each egress port, the achieved throughput rate and metrics at the egress port queue for each ingress port stream

Example:

- Egress Port 1: throughput and metrics for ingress streams 1-n

- Egress Port n: throughput and metrics for ingress streams 1-n

6.2.2.3. Multiple Queues per Port, All Ports Active

Test Summary:
Traffic from multiple ingress physical ports is directed to all queues of each egress physical port. This will cause oversubscription on the egress physical ports. Also, the same amount of traffic is directed to each egress physical port.

The benchmarking methodologies specified in Sections 6.2.1.1 (stateless) and 6.2.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. For a given priority, each ingress physical port should get a fair share of the egress physical-port bandwidth. Additionally, each egress physical port should receive the same amount of traffic.

Since this is a capacity test, the configuration and report results format (see Sections 6.2.1.1 and 6.2.1.2) MUST also include:

Configuration:

- The number of physical ingress ports active during the test

- The classification marking (DSCP, VLAN, etc.) for each physical ingress port

- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port

Report Results:

- For each egress port, the achieved throughput rate and metrics at each egress port queue for each ingress port stream

Example:

- Egress Port 1, SP Queue: throughput and metrics for ingress streams 1-n

- Egress Port 2, WFQ: throughput and metrics for ingress streams 1-n

...

- Egress Port n, SP Queue: throughput and metrics for ingress streams 1-n

- Egress Port n, WFQ: throughput and metrics for ingress streams 1-n

6.3. Shaper Tests

Like a queue, a traffic shaper is memory based, but with the added intelligence of an active traffic scheduler. The same concepts as those described in Section 6.2 (queue testing) can be applied to testing a network device shaper.

Again, the tests are divided into two sections: individual shaper benchmark tests and then full-capacity shaper benchmark tests.

6.3.1. Shaper Individual Tests

A traffic shaper generally has three (3) components that can be configured:

- Ingress Queue bytes

- Shaper Rate (SR), bps

- Burst Committed (Bc) and Burst Excess (Be), bytes

The Ingress Queue holds burst traffic, and the shaper then meters traffic out of the egress port according to the SR and Bc/Be parameters. Shapers generally transmit into policers, so the idea is for the emitted traffic to conform to the policer’s limits.

6.3.1.1. Testing Shaper with Stateless Traffic

Objective:
Test a shaper by transmitting stateless traffic bursts into the shaper ingress port and verifying that the egress traffic is shaped according to the shaper traffic profile.

Test Summary:
The stateless traffic must be burst into the DUT ingress port and not exceed the Ingress Queue. The burst can be a single burst or multiple bursts. If multiple bursts are transmitted, then the transmission interval (Ti) must be large enough so that the SR is not exceeded. An example will clarify single-burst and multiple-burst test cases.

In this example, the shaper's ingress and egress ports are both full-duplex Gigabit Ethernet. The Ingress Queue is configured to be 512,000 bytes, the SR = 50 Mbps, and both Bc and Be are configured to be 32,000 bytes. For a single-burst test, the transmitting test device would burst 512,000 bytes maximum into the ingress port and then stop transmitting.

If a multiple-burst test is to be conducted, then the burst bytes divided by the transmission interval between the 512,000-byte bursts must not exceed the SR. The transmission interval (Ti) must adhere to a formula similar to the formula described in Section 6.2.1.1 for queues, namely:

Ti = Ingress Queue * 8 / SR

For the example from the previous paragraph, the Ti between bursts must be greater than 82 milliseconds (512,000 bytes * 8 / 50,000,000 bps). This yields an average rate of 50 Mbps so that an Ingress Queue would not overflow.

Test Metrics:
The metrics defined in Section 4.1 (LP, OOS, PDV, SR, SBB, and SBI) SHALL be measured at the egress port and recorded.

Procedure:
1. Configure the DUT shaper ingress QL and shaper egress rate parameters (SR, Bc, Be).

2. Configure the tester to generate a stateless traffic burst equal to QL and an interval equal to Ti (QL in bits/BB).

3. Generate bursts of QL traffic into the DUT, and measure the metrics defined in Section 4.1 (LP, OOS, PDV, SR, SBB, and SBI) at the egress port and across the entire Td (default 30-second duration).

Reporting Format:
The Shaper Stateless Traffic individual report MUST contain all results for each QL/SR test run. A recommended format is as follows:

****

Test Configuration Summary: Tr, Td

DUT Configuration Summary: Ingress Burst Rate, QL, SR

The results table should contain entries for each test run, as follows (Test #1 to Test #Tr):

- LP, OOS, PDV, SR, SBB, and SBI

****

6.3.1.2. Testing Shaper with Stateful Traffic

Objective:
Test a shaper by transmitting stateful traffic bursts into the shaper ingress port and verifying that the egress traffic is shaped according to the shaper traffic profile.

Test Summary:
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 setup configuration in Section 1.2.

The BDP of the TCP test traffic must be calculated as described in Section 6.2.1.2. To properly stress network buffers and the traffic-shaping function, the TCP window size (which is the minimum of the TCP RWND and sender socket) should be greater than the BDP, which will stress the shaper. BDP factors of 1.1 to 1.5 are recommended, but the values are left to the discretion of the tester and should be documented.

The cumulative TCP window sizes* (RWND at the receiving end and CWND at the transmitting end) equates to the TCP window size* for each connection, multiplied by the number of connections.

* As described in Section 3 of [RFC6349], the SSB MUST be large enough to fill the BDP.

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

Two types of TCP tests MUST be performed: the Bulk Transfer Test and the Micro Burst Test Pattern, as documented in Appendix B. The Bulk Transfer Test only bursts during the TCP Slow Start (or Congestion Avoidance) state, while the Micro Burst Test Pattern emulates application-layer bursting, which may occur any time during the TCP connection.

Other types of tests SHOULD include the following: simple web sites, complex web sites, business applications, email, and SMB/CIFS file copy (all of which are also documented in Appendix B).

Test Metrics:

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

Procedure:

1. Configure the DUT shaper ingress QL and shaper egress rate parameters (SR, Bc, Be).

2. Configure the test generator* with a profile of an emulated application traffic mixture.

   - The application mixture MUST be defined in terms of percentage of the total bandwidth to be tested.

   - The rate of transmission for each application within the mixture MUST also be configurable.

   * To ensure repeatable results, the test generator MUST be capable of generating precise TCP test patterns for each application specified.

3. Generate application traffic between the ingress (client side) and egress (server side) ports of the DUT, and measure the metrics (TTPET, TCP Efficiency, and Buffer Delay) per application stream and at the ingress and egress ports (across the entire Td, default 30-second duration).

Reporting Format:
The Shaper Stateful Traffic individual report MUST contain all results for each traffic scheduler and QL/SR test run. A recommended format is as follows:

****

Test Configuration Summary: Tr, Td

DUT Configuration Summary: Ingress Burst Rate, QL, SR

Application Mixture and Intensities: These are the percentages configured for each application type.

The results table should contain entries for each test run, with minimum, maximum, and average per application session, as follows (Test #1 to Test #Tr):

- Throughput (bps) and TTPET for each application session

- Bytes In and Bytes Out for each application session

- TCP Efficiency and Buffer Delay for each application session

****

6.3.2. Shaper Capacity Tests

Objective:
The intent of these scalability tests is to verify shaper performance in a scaled environment with shapers active on multiple queues on multiple egress physical ports. These tests will benchmark the maximum number of shapers as specified by the device manufacturer.

The following sections provide the specific test scenarios, procedures, and reporting formats for each shaper capacity test.

6.3.2.1. Single Queue Shaped, All Physical Ports Active

Test Summary:

The first shaper capacity test involves per-port shaping with all physical ports active. Traffic from multiple ingress physical ports is directed to the same egress physical port. This will cause oversubscription on the egress physical port. Also, the same amount of traffic is directed to each egress physical port.

The benchmarking methodologies specified in Sections 6.3.1.1 (stateless) and 6.3.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. Since this is a capacity test, the configuration and report results format (see Section 6.3.1) MUST also include:

Configuration:

- The number of physical ingress ports active during the test
- The classification marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port
- The shaped egress port shaper parameters (QL, SR, Bc, Be)

Report Results:

- For each active egress port, the achieved throughput rate and shaper metrics for each ingress port traffic stream

Example:

- Egress Port 1: throughput and metrics for ingress streams 1-n
- Egress Port n: throughput and metrics for ingress streams 1-n

6.3.2.2. All Queues Shaped, Single Port Active

Test Summary:

The second shaper capacity test is conducted with all queues actively shaping on a single physical port. The benchmarking methodology described in the per-port shaping test (Section 6.3.2.1) serves as the foundation for this. Additionally, each of the SP queues on the egress physical port is configured with a shaper. For the highest-priority queue, the

maximum amount of bandwidth available is limited by the bandwidth of the shaper. For the lower-priority queues, the maximum amount of bandwidth available is limited by the bandwidth of the shaper and traffic in higher-priority queues.

The benchmarking methodologies specified in Sections 6.3.1.1 (stateless) and 6.3.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. Since this is a capacity test, the configuration and report results format (see Section 6.3.1) MUST also include:

Configuration:

- The number of physical ingress ports active during the test

- The classification marking (DSCP, VLAN, etc.) for each physical ingress port

- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port

- For the active egress port, each of the following shaper queue parameters: QL, SR, Bc, Be

Report Results:

- For each queue of the active egress port, the achieved throughput rate and shaper metrics for each ingress port traffic stream

Example:

- Egress Port High-Priority Queue: throughput and metrics for ingress streams 1-n

- Egress Port Lower-Priority Queue: throughput and metrics for ingress streams 1-n

6.3.2.3. All Queues Shaped, All Ports Active

Test Summary:

For the third shaper capacity test (which is a combination of the tests listed in Sections 6.3.2.1 and 6.3.2.2), all queues will be actively shaping and all physical ports active.

The benchmarking methodologies specified in Sections 6.3.1.1 (stateless) and 6.3.1.2 (stateful) (procedure, metrics, and reporting format) should be applied here. Since this is a capacity test, the configuration and report results format (see Section 6.3.1) MUST also include:

Configuration:

- The number of physical ingress ports active during the test
- The classification marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port
- For each of the active egress ports: shaper port parameters and per-queue parameters (QL, SR, Bc, Be)

Report Results:

- For each queue of each active egress port, the achieved throughput rate and shaper metrics for each ingress port traffic stream

Example:

- Egress Port 1, High-Priority Queue: throughput and metrics for ingress streams 1-n
- Egress Port 1, Lower-Priority Queue: throughput and metrics for ingress streams 1-n

...

- Egress Port n, High-Priority Queue: throughput and metrics for ingress streams 1-n
- Egress Port n, Lower-Priority Queue: throughput and metrics for ingress streams 1-n

6.4. Concurrent Capacity Load Tests

As mentioned in Section 3 of this document, it is impossible to specify the various permutations of concurrent traffic management functions that should be tested in a device for capacity testing. However, some profiles are listed below that may be useful for testing multiple configurations of traffic management functions:

- Policers on ingress and queuing on egress

- Policers on ingress and shapers on egress (not intended for a flow to be policed and then shaped; these would be two different flows tested at the same time)

The test procedures and reporting formats from Sections 6.1, 6.2, and 6.3 may be modified to accommodate the capacity test profile.

7. Security Considerations

Documents of this type do not directly affect the security of the Internet or of corporate networks as long as benchmarking is not performed on devices or systems connected to production networks.

Further, benchmarking is performed on a "black box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

8. References

8.1. Normative References

[3GPP2-C_R1002-A]
3rd Generation Partnership Project 2, "cdma2000 Evaluation Methodology", Version 1.0, Revision A, May 2009, <http://www.3gpp2.org/public_html/specs/C.R1002-A_v1.0_Evaluation_Methodology.pdf>.

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <http://www.rfc-editor.org/info/rfc1242>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <http://www.rfc-editor.org/info/rfc2544>.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <http://www.rfc-editor.org/info/rfc2680>.

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

[RFC4115] Aboul-Magd, O. and S. Rabie, "A Differentiated Service Two-Rate, Three-Color Marker with Efficient Handling of in-Profile Traffic", RFC 4115, DOI 10.17487/RFC4115, July 2005, <http://www.rfc-editor.org/info/rfc4115>.

[RFC4689] Poretsky, S., Perser, J., Erramilli, S., and S. Khurana, "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, DOI 10.17487/RFC4689, October 2006, <http://www.rfc-editor.org/info/rfc4689>.

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

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <http://www.rfc-editor.org/info/rfc5481>.

[RFC6349] Constantine, B., Forget, G., Geib, R., and R. Schrage, "Framework for TCP Throughput Testing", RFC 6349, DOI 10.17487/RFC6349, August 2011, <http://www.rfc-editor.org/info/rfc6349>.

[RFC6703] Morton, A., Ramachandran, G., and G. Maguluri, "Reporting IP Network Performance Metrics: Different Points of View", RFC 6703, DOI 10.17487/RFC6703, August 2012, <http://www.rfc-editor.org/info/rfc6703>.

[SPECweb2009] Standard Performance Evaluation Corporation (SPEC), "SPECweb2009 Release 1.20 Benchmark Design Document", April 2010, <https://www.spec.org/web2009/docs/design/SPECweb2009_Design.html>.

8.2. Informative References

[CA-Benchmark] Hamilton, M. and S. Banks, "Benchmarking Methodology for Content-Aware Network Devices", Work in Progress, draft-ietf-bmwg-ca-bench-meth-04, February 2013.

[CoDel] Nichols, K., Jacobson, V., McGregor, A., and J. Iyengar, "Controlled Delay Active Queue Management", Work in Progress, draft-ietf-aqm-codel-01, April 2015.

[MEF-10.3] Metro Ethernet Forum, "Ethernet Services Attributes Phase 3", MEF 10.3, October 2013, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_10.3.pdf>.

[MEF-12.2] Metro Ethernet Forum, "Carrier Ethernet Network Architecture Framework -- Part 2: Ethernet Services Layer", MEF 12.2, May 2014, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_12.2.pdf>.

[MEF-14] Metro Ethernet Forum, "Abstract Test Suite for Traffic Management Phase 1", MEF 14, November 2005, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_14.pdf>.

[MEF-19] Metro Ethernet Forum, "Abstract Test Suite for UNI Type 1", MEF 19, April 2007, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_19.pdf>.

[MEF-26.1] Metro Ethernet Forum, "External Network Network Interface (ENNI) - Phase 2", MEF 26.1, January 2012, <http://www.mef.net/Assets/Technical_Specifications/PDF/MEF_26.1.pdf>.

[MEF-37] Metro Ethernet Forum, "Abstract Test Suite for ENNI", MEF 37, January 2012, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_37.pdf>.

[PIE] Pan, R., Natarajan, P., Baker, F., White, G., VerSteeg, B., Prabhu, M., Piglione, C., and V. Subramanian, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", Work in Progress, draft-ietf-aqm-pie-02, August 2015.

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <http://www.rfc-editor.org/info/rfc2697>.

[RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, DOI 10.17487/RFC2698, September 1999, <http://www.rfc-editor.org/info/rfc2698>.

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

Appendix A. Open Source Tools for Traffic Management Testing

This framework specifies that stateless and stateful behaviors SHOULD both be tested. Some open source tools that can be used to accomplish many of the tests proposed in this framework are iperf, netperf (with netperf-wrapper), the "uperf" tool, Tmix, TCP-incast-generator, and D-ITG (Distributed Internet Traffic Generator).

iperf can generate UDP-based 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 unidirectional and bidirectional concurrent testing are supported.

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

iperf (TCP mode), TCP-incast-generator, and D-ITG can be used for stateful traffic testing to test bulk transfer traffic. The TCP window size (which is actually the SSB), number of connections, packet size, TCP port, and test duration can be controlled. A report of bytes transmitted and throughput achieved is provided by the iperf sender, while TCP-incast-generator and D-ITG provide even more statistics.

netperf is a software application that provides network bandwidth testing between two hosts on a network. It supports UNIX domain sockets, TCP, SCTP, and UDP via BSD Sockets. netperf provides a number of predefined tests, e.g., to measure bulk (unidirectional) data transfer or request/response performance (http://en.wikipedia.org/wiki/Netperf). netperf-wrapper is a Python script that runs multiple simultaneous netperf instances and aggregates the results.

uperf uses a description (or model) of an application mixture. It generates the load according to the model descriptor. uperf is more flexible than netperf in its ability to generate request/response application behavior within a single TCP connection. The application model descriptor can be based on empirical data, but at the time of this writing, the import of packet captures is not directly supported.

Tmix is another application traffic emulation tool. It uses packet captures directly to create the traffic profile. The packet trace is "reverse compiled" into a source-level characterization, called a "connection vector", of each TCP connection present in the trace. While most widely used in ns2 simulation environments, Tmix also runs on Linux hosts.

The traffic generation capabilities of these open source tools facilitate the emulation of the TCP test patterns discussed in Appendix B.

Appendix B. Stateful TCP Test Patterns

This framework recommends at a minimum the following TCP test patterns, since they are representative of real-world application traffic (Section 5.2.1 describes some methods to derive other application-based TCP test patterns).

- Bulk Transfer: Generate concurrent TCP connections whose aggregate number of in-flight data bytes would fill the BDP. Guidelines from [RFC6349] are used to create this TCP traffic pattern.

- Micro Burst: Generate precise burst patterns within a single TCP connection or multiple TCP connections. The idea is for TCP to establish equilibrium and then burst application bytes at defined sizes. The test tool must allow the burst size and burst time interval to be configurable.

- Web Site Patterns: The HTTP traffic model shown in Table 4.1.3-1 of [3GPP2-C_R1002-A] demonstrates a way to develop these TCP test patterns. In summary, the HTTP traffic model consists of the following parameters:

  - Main object size (Sm)

  - Embedded object size (Se)

  - Number of embedded objects per page (Nd)

  - Client processing time (Tcp)

  - Server processing time (Tsp)

Web site test patterns are illustrated with the following examples:

- Simple web site: Mimic the request/response and object download behavior of a basic web site (small company).

- Complex web site: Mimic the request/response and object download behavior of a complex web site (eCommerce site).

Referencing the HTTP traffic model parameters, the following table was derived (by analysis and experimentation) for simple web site and complex web site TCP test patterns:

Parameter	Simple Web Site	Complex Web Site
Main object size (Sm)	Ave. = 10KB Min. = 100B Max. = 500KB	Ave. = 300KB Min. = 50KB Max. = 2MB
Embedded object size (Se)	Ave. = 7KB Min. = 50B Max. = 350KB	Ave. = 10KB Min. = 100B Max. = 1MB
Number of embedded objects per page (Nd)	Ave. = 5 Min. = 2 Max. = 10	Ave. = 25 Min. = 10 Max. = 50
Client processing time (Tcp)*	Ave. = 3s Min. = 1s Max. = 10s	Ave. = 10s Min. = 3s Max. = 30s
Server processing time (Tsp)*	Ave. = 5s Min. = 1s Max. = 15s	Ave. = 8s Min. = 2s Max. = 30s

* The client and server processing time is distributed across the transmission/receipt of all of the main and embedded objects.

To be clear, the parameters in this table are reasonable guidelines for the TCP test pattern traffic generation. The test tool can use fixed parameters for simpler tests and mathematical distributions for more complex tests. However, the test pattern must be repeatable to ensure that the benchmark results can be reliably compared.

- Interactive Patterns: While web site patterns are interactive to a degree, they mainly emulate the downloading of web sites of varying complexity. Interactive patterns are more chatty in nature, since there is a lot of user interaction with the servers. Examples include business applications such as PeopleSoft and Oracle, and consumer applications such as Facebook and IM. For the interactive patterns, the packet capture technique was used to characterize some business applications and also the email application.

In summary, an interactive application can be described by the following parameters:

- Client message size (Scm)

- Number of client messages (Nc)

- Server response size (Srs)

- Number of server messages (Ns)

- Client processing time (Tcp)

- Server processing time (Tsp)

- File size upload (Su)*

- File size download (Sd)*

* The file size parameters account for attachments uploaded or downloaded and may not be present in all interactive applications.

Again using packet capture as a means to characterize, the following table reflects the guidelines for simple business applications, complex business applications, eCommerce, and email Send/Receive:

Parameter	Simple Business Application	Complex Business Application	eCommerce*	Email
Client message size (Scm)	Ave. = 450B Min. = 100B Max. = 1.5KB	Ave. = 2KB Min. = 500B Max. = 100KB	Ave. = 1KB Min. = 100B Max. = 50KB	Ave. = 200B Min. = 100B Max. = 1KB
Number of client messages (Nc)	Ave. = 10 Min. = 5 Max. = 25	Ave. = 100 Min. = 50 Max. = 250	Ave. = 20 Min. = 10 Max. = 100	Ave. = 10 Min. = 5 Max. = 25
Client processing time (Tcp)**	Ave. = 10s Min. = 3s Max. = 30s	Ave. = 30s Min. = 3s Max. = 60s	Ave. = 15s Min. = 5s Max. = 120s	Ave. = 5s Min. = 3s Max. = 45s
Server response size (Srs)	Ave. = 2KB Min. = 500B Max. = 100KB	Ave. = 5KB Min. = 1KB Max. = 1MB	Ave. = 8KB Min. = 100B Max. = 50KB	Ave. = 200B Min. = 150B Max. = 750B
Number of server messages (Ns)	Ave. = 50 Min. = 10 Max. = 200	Ave. = 200 Min. = 25 Max. = 1000	Ave. = 100 Min. = 15 Max. = 500	Ave. = 15 Min. = 5 Max. = 40
Server processing time (Tsp)**	Ave. = 0.5s Min. = 0.1s Max. = 5s	Ave. = 1s Min. = 0.5s Max. = 20s	Ave. = 2s Min. = 1s Max. = 10s	Ave. = 4s Min. = 0.5s Max. = 15s
File size upload (Su)	Ave. = 50KB Min. = 2KB Max. = 200KB	Ave. = 100KB Min. = 10KB Max. = 2MB	Ave. = N/A Min. = N/A Max. = N/A	Ave. = 100KB Min. = 20KB Max. = 10MB
File size download (Sd)	Ave. = 50KB Min. = 2KB Max. = 200KB	Ave. = 100KB Min. = 10KB Max. = 2MB	Ave. = N/A Min. = N/A Max. = N/A	Ave. = 100KB Min. = 20KB Max. = 10MB

* eCommerce used a combination of packet capture techniques and reference traffic flows as described in [SPECweb2009].

** The client and server processing time is distributed across the transmission/receipt of all of the messages. The client processing time consists mainly of the delay between user interactions (not machine processing).

Again, the parameters in this table are the guidelines for the TCP test pattern traffic generation. The test tool can use fixed parameters for simpler tests and mathematical distributions for more complex tests. However, the test pattern must be repeatable to ensure that the benchmark results can be reliably compared.

- SMB/CIFS file copy: Mimic a network file copy, both read and write. As opposed to FTP, which is a bulk transfer and is only flow-controlled via TCP, SMB/CIFS divides a file into application blocks and utilizes application-level handshaking in addition to TCP flow control.

In summary, an SMB/CIFS file copy can be described by the following parameters:

- Client message size (Scm)

- Number of client messages (Nc)

- Server response size (Srs)

- Number of server messages (Ns)

- Client processing time (Tcp)

- Server processing time (Tsp)

- Block size (Sb)

The client and server messages are SMB control messages. The block size is the data portion of the file transfer.

Again using packet capture as a means to characterize, the following table reflects the guidelines for SMB/CIFS file copy:

Parameter	SMB/CIFS File Copy
Client message size (Scm)	Ave. = 450B Min. = 100B Max. = 1.5KB
Number of client messages (Nc)	Ave. = 10 Min. = 5 Max. = 25
Client processing time (Tcp)	Ave. = 1ms Min. = 0.5ms Max. = 2
Server response size (Srs)	Ave. = 2KB Min. = 500B Max. = 100KB
Number of server messages (Ns)	Ave. = 10 Min. = 10 Max. = 200
Server processing time (Tsp)	Ave. = 1ms Min. = 0.5ms Max. = 2ms
Block size (Sb)*	Ave. = N/A Min. = 16KB Max. = 128KB

* Depending upon the tested file size, the block size will be transferred "n" number of times to complete the example. An example would be a 10 MB file test and 64 KB block size. In this case, 160 blocks would be transferred after the control channel is opened between the client and server.

Acknowledgments

We would like to thank Al Morton for his continuous review and invaluable input to this document. We would also like to thank Scott Bradner for providing guidance early in this document's conception, in the area of the benchmarking scope of traffic management functions. Additionally, we would like to thank Tim Copley for his original input, as well as David Taht, Gory Erg, and Toke Hoiland-Jorgensen for their review and input for the AQM group. Also, for the formal reviews of this document, we would like to thank Gilles Forget, Vijay Gurbani, Reinhard Schrage, and Bhuvaneswaran Vengainathan.

Authors' Addresses

Barry Constantine
JDSU, Test and Measurement Division
Germantown, MD 20876-7100
United States

Phone: +1-240-404-2227
Email: barry.constantine@jdsu.com

Ram (Ramki) Krishnan
Dell Inc.
Santa Clara, CA 95054
United States

Phone: +1-408-406-7890
Email: ramkri123@gmail.com