Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 7, 2011

B. Constantine
JDSU
G. Forget
Bell Canada (Ext. Consultant)
Rudiger Geib
Deutsche Telekom
Reinhard Schrage
Schrage Consulting

December 7, 2010

Framework for TCP Throughput Testing
draft-ietf-ippm-tcp-throughput-tm-09.txt

Abstract

This framework describes a methodology for measuring end-to-end TCP throughput performance in a managed IP network. The intention is to provide a practical methodology to validate TCP layer performance. The goal is to provide a better indication of the user experience. In this framework, various TCP and IP parameters are identified and should be tested as part of a managed IP network.

Requirements Language

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].

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on June 7, 2011.

Table of Contents

1. Introduction

Network providers are coming to the realization that Layer 2/3 testing is not enough to adequately ensure end-user's satisfaction. An SLA (Service Level Agreement) is provided to business customers and is generally based upon Layer 2/3 criteria such as access rate, latency, packet loss and delay variations. On the other hand, measuring TCP throughput provides meaningful results with respect to user experience. Thus, the network provider community desires to measure IP network throughput performance at the TCP layer.

Additionally, business enterprise customers seek to conduct repeatable TCP throughput tests between locations. Since these enterprises rely on the networks of the providers, a common test methodology with predefined metrics will benefit both parties.

Note that the primary focus of this methodology is managed business class IP networks; i.e. those Ethernet terminated services for which businesses are provided an SLA from the network provider. End-users with "best effort" access between locations can use this methodology, but this framework and its metrics are intended to be used in a predictable managed IP service environment.

So the intent behind this document is to define a methodology for testing sustained TCP layer performance. In this document, the maximum achievable TCP Throughput is that amount of data per unit time that TCP transports when trying to reach Equilibrium, i.e. after the initial slow start and congestion avoidance phases.

TCP uses a congestion window, (TCP CWND), to determine how many packets it can send at one time. The network path bandwidth delay product (BDP) determines the ideal TCP CWND. With the help of slow start and congestion avoidance mechanisms, TCP probes the network path. So up to the bandwidth limit, a larger TCP CWND permits a higher throughput. And up to local host limits, TCP "Slow Start" and "Congestion Avoidance" algorithms together will determine the TCP CWND size. The Maximum TCP CWND size is also tributary to the buffer space allocated by the kernel for each socket. For each socket, there is a default buffer size that can be changed by the program using a system library called just before opening the socket. There is also a kernel enforced maximum buffer size. This buffer size can be adjusted at both ends of the socket (send and receive). In order to obtain the maximum throughput, it is critical to use optimal TCP Send and Receive Socket Buffer sizes.

There are many variables to consider when conducting a TCP throughput test, but this methodology focuses on:
- RTT and Bottleneck BW
- Ideal TCP Receive Window (Ideal Receive Socket Buffer)

- Ideal Send Socket Buffer
- TCP Congestion Window (TCP CWND)
- Path MTU and Maximum Segment Size (MSS)
- Single Connection and Multiple Connections testing

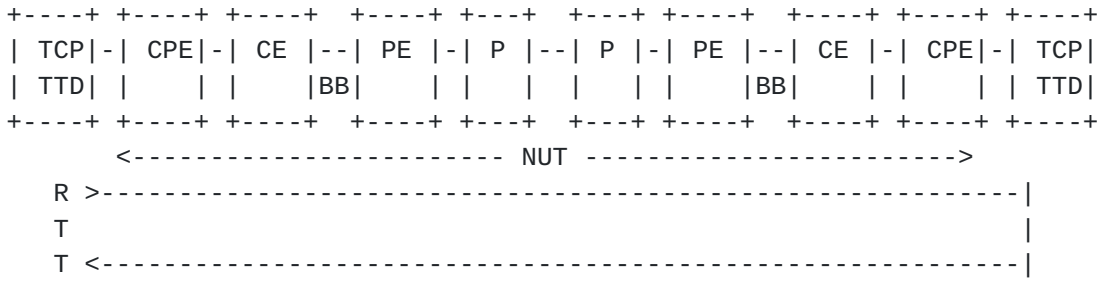
This methodology proposes TCP testing that should be performed in addition to traditional Layer 2/3 type tests. Layer 2/3 tests are required to verify the integrity of the network before conducting TCP tests. Examples include iperf (UDP mode) or manual packet layer test techniques where packet throughput, loss, and delay measurements are conducted. When available, standardized testing similar to RFC 2544 [RFC2544] but adapted for use in operational networks may be used. Note: RFC 2544 was never meant to be used outside a lab environment.

The following 2 sections provide a general overview of the test methodology.

1.1 Terminology

Common terminologies used in the test methodology are:

- TCP Throughput Test Device (TCP TTD), refers to compliant TCP host that generates traffic and measures metrics as defined in this methodology. i.e. a dedicated communications test instrument.
- Customer Provided Equipment (CPE), refers to customer owned equipment (routers, switches, computers, etc.)
- Customer Edge (CE), refers to provider owned demarcation device.
- Provider Edge (PE), refers to provider's distribution equipment.
- Bottleneck Bandwidth (BB), lowest bandwidth along the complete path. Bottleneck Bandwidth and Bandwidth are used synonymously in this document. Most of the time the Bottleneck Bandwidth is in the access portion of the wide area network (CE - PE)
- Provider (P), refers to provider core network equipment.
- Network Under Test (NUT), refers to the tested IP network path.
- Round-Trip Time (RTT), refers to Layer 4 back and forth delay.



Note that the NUT may consist of a variety of devices including but not limited to, load balancers, proxy servers or WAN acceleration devices. The detailed topology of the NUT should be well understood when conducting the TCP throughput tests, although this methodology makes no attempt to characterize specific network architectures.

1.2 Test Set-up

This methodology is intended for operational and managed IP networks.

A multitude of network architectures and topologies can be tested. The above set-up diagram is very general and it only illustrates the segmentation within end-user and network provider domains.

2. Scope and Goals of this Methodology

Before defining the goals, it is important to clearly define the areas that are out-of-scope.

- This methodology is not intended to predict the TCP throughput during the transient stages of a TCP connection, such as the initial slow start.

- This methodology is not intended to definitively benchmark TCP implementations of one OS to another, although some users may find some value in conducting qualitative experiments.

- This methodology is not intended to provide detailed diagnosis of problems within end-points or within the network itself as related to non-optimal TCP performance, although a results interpretation section for each test step may provide insight in regards with potential issues.

- This methodology does not propose to operate permanently with high measurement loads. TCP performance and optimization within operational networks may be captured and evaluated by using data from the "TCP Extended Statistics MIB" [RFC4898].

- This methodology is not intended to measure TCP throughput as part of an SLA, or to compare the TCP performance between service providers or to compare between implementations of this methodology in dedicated communications test instruments.

In contrast to the above exclusions, a primary goal is to define a method to conduct a practical, end-to-end assessment of sustained TCP performance within a managed business class IP network. Another key goal is to establish a set of "best practices" that a non-TCP expert should apply when validating the ability of a managed network to carry end-user TCP applications.

Other specific goals are to :

- Provide a practical test approach that specifies IP hosts configurable TCP parameters such as TCP Receive Window size, Socket Buffer size, MSS (Maximum Segment Size), number of connections, and how these affect the outcome of TCP performance over a network. See section 3.3.3.

- Provide specific test conditions like link speed, RTT, TCP Receive Window size, Socket Buffer size and maximum achievable TCP throughput when trying to reach TCP Equilibrium. For guideline purposes, provide examples of test conditions and their maximum achievable TCP throughput. Section 2.1 provides specific details concerning the definition of TCP Equilibrium within this methodology while section 3

provides specific test conditions with examples.

Note that some TCP/IP stack implementations are using Receive Window Auto-Tuning and cannot be adjusted until this feature is disabled.

- Define three (3) basic metrics to compare the performance of TCP connections under various network conditions. See section 3.3.2.

- In test situations where the recommended procedure does not yield the maximum achievable TCP throughput results, this methodology provides some possible areas within the end host or the network that should be considered for investigation. Although again, this methodology is not intended to provide a detailed diagnosis on these issues. See section 3.3.5.

2.1 TCP Equilibrium

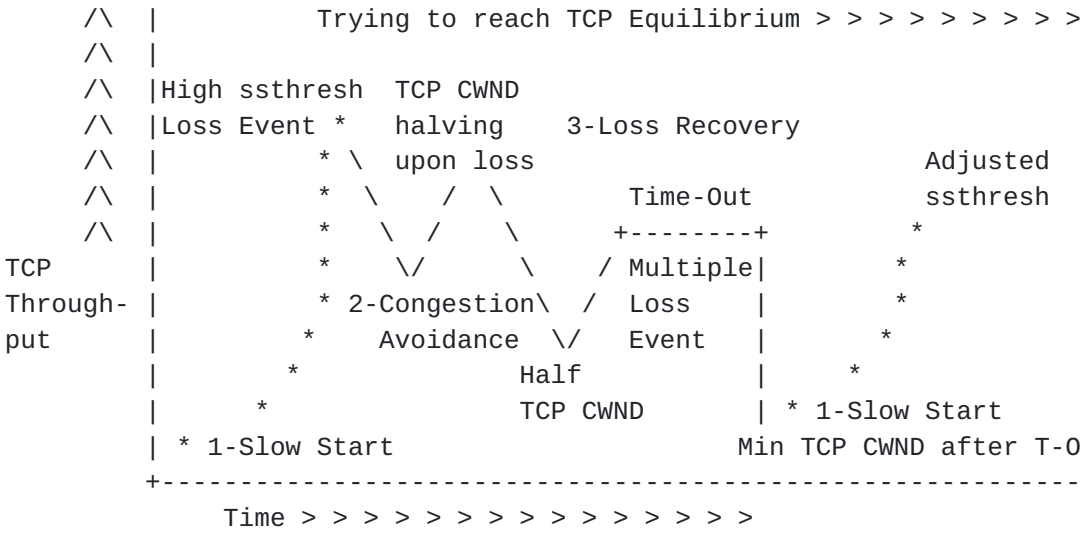
TCP connections have three (3) fundamental congestion window phases :

1 - The Slow Start phase, which occurs at the beginning of a TCP transmission or after a retransmission time out.

2 - The Congestion Avoidance phase, during which TCP ramps up to establish the maximum attainable throughput on an end-to-end network path. Retransmissions are a natural by-product of the TCP congestion avoidance algorithm as it seeks to achieve maximum throughput.

3 - The Loss Recovery phase, which could include Fast Retransmit (Tahoe) or Fast Recovery (Reno & New Reno). When packet loss occurs, Congestion Avoidance phase transitions either to Fast Retransmission or Fast Recovery depending upon TCP implementations. If a Time-Out occurs, TCP transitions back to the Slow Start phase.

The following diagram depicts these 3 phases.



Note : ssthresh = Slow Start threshold.

A well tuned and managed IP network with appropriate TCP adjustments in it's IP hosts and applications should perform very close to TCP

Equilibrium and to the BB (Bottleneck Bandwidth).

This TCP methodology provides guidelines to measure the maximum achievable TCP throughput or maximum TCP sustained rate obtained after TCP CWND has stabilized to an optimal value. All maximum achievable TCP throughputs specified in section 3 are with respect to this condition.

It is important to clarify the interaction between the sender's Send Socket Buffer and the receiver's advertised TCP Receive Window. TCP test programs such as iperf, ttcp, etc. allow the sender to control the quantity of TCP Bytes transmitted and unacknowledged (in-flight), commonly referred to as the Send Socket Buffer. This is done independently of the TCP Receive Window size advertised by the receiver. Implications to the capabilities of the Throughput Test Device (TTD) are covered at the end of section 3.

3. TCP Throughput Testing Methodology

As stated earlier in section 1, it is considered best practice to verify the integrity of the network by conducting Layer2/3 tests such as [RFC2544] or other methods of network stress tests. Although, it is important to mention here that RFC 2544 was never meant to be used outside a lab environment.

If the network is not performing properly in terms of packet loss, jitter, etc. then the TCP layer testing will not be meaningful. A dysfunctional network will not acheive optimal TCP throughputs in regards with the available bandwidth.

TCP Throughput testing may require cooperation between the end-user customer and the network provider. In a Layer 2/3 VPN architecture, the testing should be conducted either on the CPE or on the CE device and not on the PE (Provider Edge) router.

The following represents the sequential order of steps for this testing methodology:

1. Identify the Path MTU. Packetization Layer Path MTU Discovery or PLPMTUD, [RFC4821], MUST be conducted to verify the network path MTU. Conducting PLPMTUD establishes the upper limit for the MSS to be used in subsequent steps.

2. Baseline Round Trip Time and Bandwidth. This step establishes the inherent, non-congested Round Trip Time (RTT) and the bottleneck bandwidth of the end-to-end network path. These measurements are used to provide estimates of the ideal TCP Receive Window and Send Socket Buffer sizes that SHOULD be used in subsequent test steps. These measurements reference [RFC2681] and [RFC4898] to measure RTD and the associated RTT.

3. TCP Connection Throughput Tests. With baseline measurements

of Round Trip Time and bottleneck bandwidth, single and multiple TCP connection throughput tests SHOULD be conducted to baseline network performance expectations.

4. Traffic Management Tests. Various traffic management and queuing techniques can be tested in this step, using multiple TCP connections. Multiple connections testing should verify that the network is configured properly for traffic shaping versus policing, various queuing implementations and Random Early Discards (RED).

Important to note are some of the key characteristics and considerations for the TCP test instrument. The test host may be a standard computer or a dedicated communications test instrument. In both cases, they must be capable of emulating both client and server.

The following criteria should be considered when selecting whether the TCP test host can be a standard computer or has to be a dedicated communications test instrument:

- TCP implementation used by the test host, OS version, i.e. Linux OS kernel using TCP New Reno, TCP options supported, etc. These will obviously be more important when using dedicated communications test instruments where the TCP implementation may be customized or tuned to run in higher performance hardware. When a compliant TCP TTD is used, the TCP implementation MUST be identified in the test results. The compliant TCP TTD should be usable for complete end-to-end testing through network security elements and should also be usable for testing network sections.

- More important, the TCP test host MUST be capable to generate and receive stateful TCP test traffic at the full link speed of the network under test. Stateful TCP test traffic means that the test host MUST fully implement a TCP/IP stack; this is generally a comment aimed at dedicated communications test equipments which sometimes "blast" packets with TCP headers. As a general rule of thumb, testing TCP throughput at rates greater than 100 Mbit/sec MAY require high performance server hardware or dedicated hardware based test tools.

- A compliant TCP Throughput Test Device MUST allow adjusting both Send and Receive Socket Buffer sizes. The Receive Socket Buffer MUST be large enough to accommodate the TCP Receive Window Size. Note that some TCP/IP stack implementations are using Receive Window Auto-Tuning and cannot be adjusted until this feature is disabled.

- 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 retransmission results in packet captures. Another option may be to use "TCP Extended Statistics MIB" per [RFC4898].

- The RFC4821 PLPMTUD test SHOULD be conducted with a dedicated tester which exposes the ability to run the PLPMTUD algorithm independent from the OS stack.

3.1. Determine Network Path MTU

TCP implementations should use Path MTU Discovery techniques (PMTUD). PMTUD relies on ICMP 'need to frag' messages to learn the path MTU. When a device has a packet to send which has the Don't Fragment (DF) bit in the IP header set and the packet is larger than the Maximum Transmission Unit (MTU) of the next hop, the packet is dropped and the device sends an ICMP 'need to frag' message back to the host that originated the packet. The ICMP 'need to frag' message includes the next hop MTU which PMTUD uses to tune the TCP Maximum Segment Size (MSS). Unfortunately, because many network managers completely disable ICMP, this technique does not always prove reliable.

Packetization Layer Path MTU Discovery or PLPMTUD [RFC4821] MUST then be conducted to verify the network path MTU. PLPMTUD can be used with or without ICMP. The following sections provide a summary of the PLPMTUD approach and an example using TCP. [RFC4821] specifies a search_high and a search_low parameter for the MTU. As specified in [RFC4821], 1024 Bytes is a safe value for search_low in modern networks.

It is important to determine the links overhead along the IP path, and then to select a TCP MSS size corresponding to the Layer 3 MTU. For example, if the MTU is 1024 Bytes and the TCP/IP headers are 40 Bytes, then the MSS would be set to 984 Bytes.

An example scenario is a network where the actual path MTU is 1240 Bytes. The TCP client probe MUST be capable of setting the MSS for the probe packets and could start at MSS = 984 (which corresponds to an MTU size of 1024 Bytes).

The TCP client probe would open a TCP connection and advertise the MSS as 984. Note that the client probe MUST generate these packets with the DF bit set. The TCP client probe then sends test traffic per a small default Send Socket Buffer size of ~8KBytes. It should be kept small to minimize the possibility of congesting the network, which may induce packet loss. The duration of the test should also be short (10-30 seconds), again to minimize congestive effects during the test.

In the example of a 1240 Bytes path MTU, probing with an MSS equal to 984 would yield a successful probe and the test client packets would be successfully transferred to the test server.

Also note that the test client MUST verify that the MSS advertised is indeed negotiated. Network devices with built-in Layer 4 capabilities can intercede during the connection establishment and reduce the advertised MSS to avoid fragmentation. This is certainly a desirable feature from a network perspective, but it can yield erroneous test results if the client test probe does not confirm the negotiated MSS.

The next test probe would use the search_high value and this would be set to MSS = 1460 to correspond to a 1500 Bytes MTU. In this example, the test client will retransmit based upon time-outs, since no ACKs will be received from the test server. This test probe is marked as a conclusive failure if none of the test packets are ACK'ed. If any of the test packets are ACK'ed, congestive network may be the cause and the test probe is not conclusive. Re-testing at other times of the day is recommended to further isolate.

The test is repeated until the desired granularity of the MTU is discovered. The method can yield precise results at the expense of probing time. One approach may be to reduce the probe size to half between the unsuccessful search_high and successful search_low value and raise it by half also when seeking the upper limit.

3.2. Baseline Round Trip Time and Bandwidth

Before stateful TCP testing can begin, it is important to determine the baseline Round Trip Time (non-congested inherent delay) and bottleneck bandwidth of the end-to-end network to be tested. These measurements are used to provide estimates of the ideal TCP Receive Window and Send Socket Buffer sizes that SHOULD be used in subsequent test steps.

3.2.1 Techniques to Measure Round Trip Time

Following the definitions used in section 1.1, Round Trip Time (RTT) is the elapsed time between the clocking in of the first bit of a payload sent packet to the receipt of the last bit of the corresponding Acknowledgment. Round Trip Delay (RTD) is used synonymously to twice the Link Latency. RTT measurements SHOULD use techniques defined in [RFC2681] or statistics available from MIBs defined in [RFC4898].

The RTT SHOULD be baselined during "off-peak" hours to obtain a reliable figure for inherent network latency versus additional delay caused by network buffering. When sampling values of RTT over a test interval, the minimum value measured SHOULD be used as the baseline RTT since this will most closely estimate the inherent network latency. This inherent RTT is also used to determine the Buffer Delay Percentage metric which is defined in Section 3.3.2

The following list is not meant to be exhaustive, although it summarizes some of the most common ways to determine round trip time. The desired resolution of the measurement (i.e. msec versus usec) may dictate whether the RTT measurement can be achieved with ICMP pings or by a dedicated communications test instrument with precision timers.

The objective in this section is to list several techniques in order of decreasing accuracy.

- Use test equipment on each end of the network, "looping" the far-end tester so that a packet stream can be measured back and forth from end-to-end. This RTT measurement may be compatible with delay measurement protocols specified in [RFC5357].

- Conduct packet captures of TCP test sessions using "iperf" or FTP, or other TCP test applications. By running multiple experiments, packet captures can then be analyzed to estimate RTT. It is important to note that results based upon the SYN -> SYN-ACK at the beginning of TCP sessions should be avoided since Firewalls might slow down 3 way handshakes.

- ICMP pings may also be adequate to provide round trip time estimates, provided that the packet size is factored into the estimates (i.e. pings with different packet sizes might be required). Some limitations with ICMP Ping may include msec resolution and whether the network elements are responding to pings or not. Also, ICMP is often rate-limited and segregated into different buffer queues and is not as reliable and accurate as in-band measurements.

3.2.2 Techniques to Measure end-to-end Bandwidth

There are many well established techniques available to provide estimated measures of bandwidth over a network. These measurements SHOULD be conducted in both directions of the network, especially for access networks, which may be asymmetrical. Measurements SHOULD use network capacity techniques defined in [RFC5136].

Before any TCP Throughput test can be done, a bandwidth measurement test MUST be run with stateless IP streams(not stateful TCP) in order to determine the available bandwidths in each direction. This test should obviously be performed at various intervals throughout a business day or even across a week. Ideally, the bandwidth test should produce logged outputs of the achieved bandwidths across the test interval.

3.3. TCP Throughput Tests

This methodology specifically defines TCP throughput techniques to verify sustained TCP performance in a managed business IP network, as defined in section 2.1. This section and others will define the method to conduct these sustained TCP throughput tests and guidelines for the predicted results.

With baseline measurements of round trip time and bandwidth from section 3.2, a series of single and multiple TCP connection throughput tests SHOULD be conducted to baseline network performance against expectations. The number of trials and the type of testing (single versus multiple connections) will vary according to the intention of the test. One example would be a single connection test in which the throughput achieved by large Send Socket Buffer and TCP Receive Window sizes (i.e. 256KB) is to be measured. It would be advisable to test performance at various times of the business day.

It is RECOMMENDED to run the tests in each direction independently first, then run both directions simultaneously. In each case, TCP Transfer Time, TCP Efficiency, and Buffer Delay Percentage MUST be measured in each direction. These metrics are defined in 3.3.2.

3.3.1 Calculate Ideal TCP Receive Window Size

The ideal TCP Receive Window size can be calculated from the bandwidth delay product (BDP), which is:

BDP (bits) = RTT (sec) x Bandwidth (bps)

Note that the RTT is being used as the "Delay" variable in the BDP calculations.

Then, by dividing the BDP by 8, we obtain the "ideal" TCP Receive Window size in Bytes. For optimal results, the Send Socket Buffer size must be adjusted to the same value at the opposite end of the network path.

Ideal TCP RWIN = BDP / 8

An example would be a T3 link with 25 msec RTT. The BDP would equal ~1,105,000 bits and the ideal TCP Receive Window would be ~138 KBytes.

Note that separate calculations are required on asymetrical paths. An asymetrical path example would be a 90 msec RTT ADSL line with 5Mbps downstream and 640Kbps upstream. The downstream BDP would equal ~450,000 bits while the upstream one would be only ~57,600 bits.

The following table provides some representative network Link Speeds, RTT, BDP, and their associated Ideal TCP Receive Window sizes.

Table 3.3.1: Link Speed, RTT and calculated BDP & TCP Receive Window

Link Speed* (Mbps)	RTT (ms)	BDP (bits)	Ideal TCP Receive Window (KBytes)
1.536	20	30,720	3.84
1.536	50	76,800	9.60
1.536	100	153,600	19.20
44.210	10	442,100	55.26
44.210	15	663,150	82.89
44.210	25	1,105,250	138.16
100	1	100,000	12.50
100	2	200,000	25.00
100	5	500,000	62.50
1,000	0.1	100,000	12.50
1,000	0.5	500,000	62.50
1,000	1	1,000,000	125.00
10,000	0.05	500,000	62.50
10,000	0.3	3,000,000	375.00

* Note that link speed is the bottleneck bandwidth for the NUT

The following serial link speeds are used:
- T1 = 1.536 Mbits/sec (for a B8ZS line encoding facility)
- T3 = 44.21 Mbits/sec (for a C-Bit Framing facility)

The above table illustrates the ideal TCP Receive Window size.
If a smaller TCP Receive Window is used, then the TCP Throughput
is not optimal. To calculate the TCP Throughput, the following
formula is used: TCP Throughput = TCP RWIN X 8 / RTT

An example could be a 100 Mbps IP path with 5 ms RTT and a TCP
Receive Window size of 16KB, then:

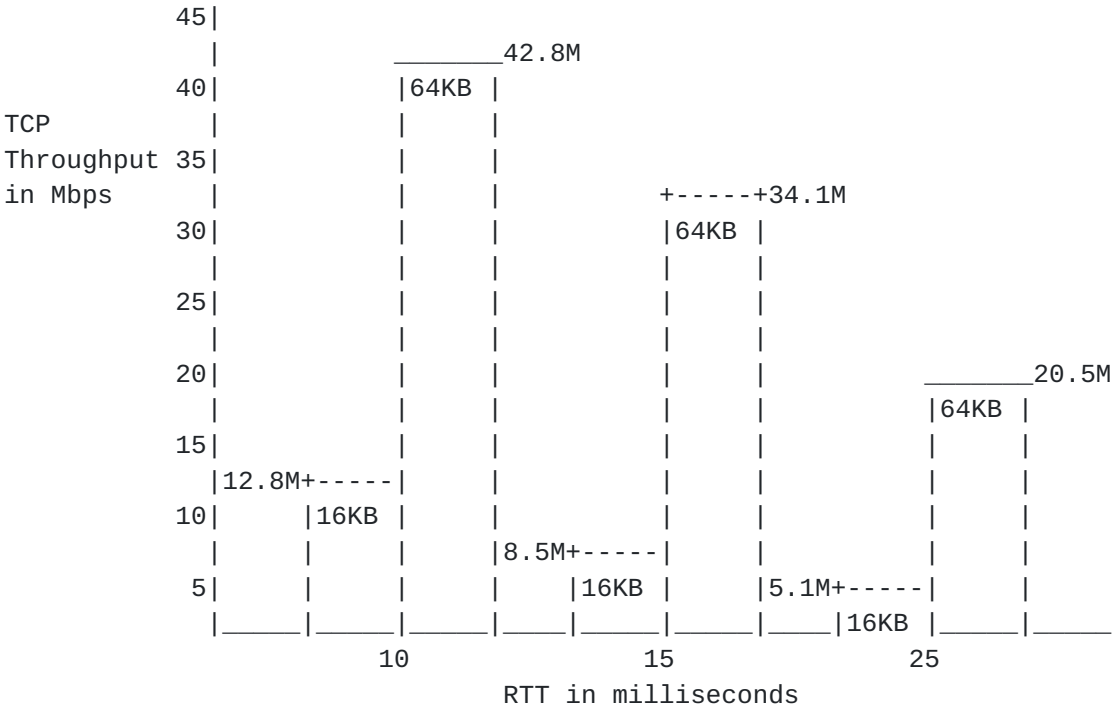
TCP Throughput = 16 KBytes X 8 bits / 5 ms.
TCP Throughput = 128,000 bits / 0.005 sec.
TCP Throughput = 25.6 Mbps.

Another example for a T3 using the same calculation formula is
illustrated on the next page:
TCP Throughput = TCP RWIN X 8 / RTT.
TCP Throughput = 16 KBytes X 8 bits / 10 ms.
TCP Throughput = 128,000 bits / 0.01 sec.
TCP Throughput = 12.8 Mbps.

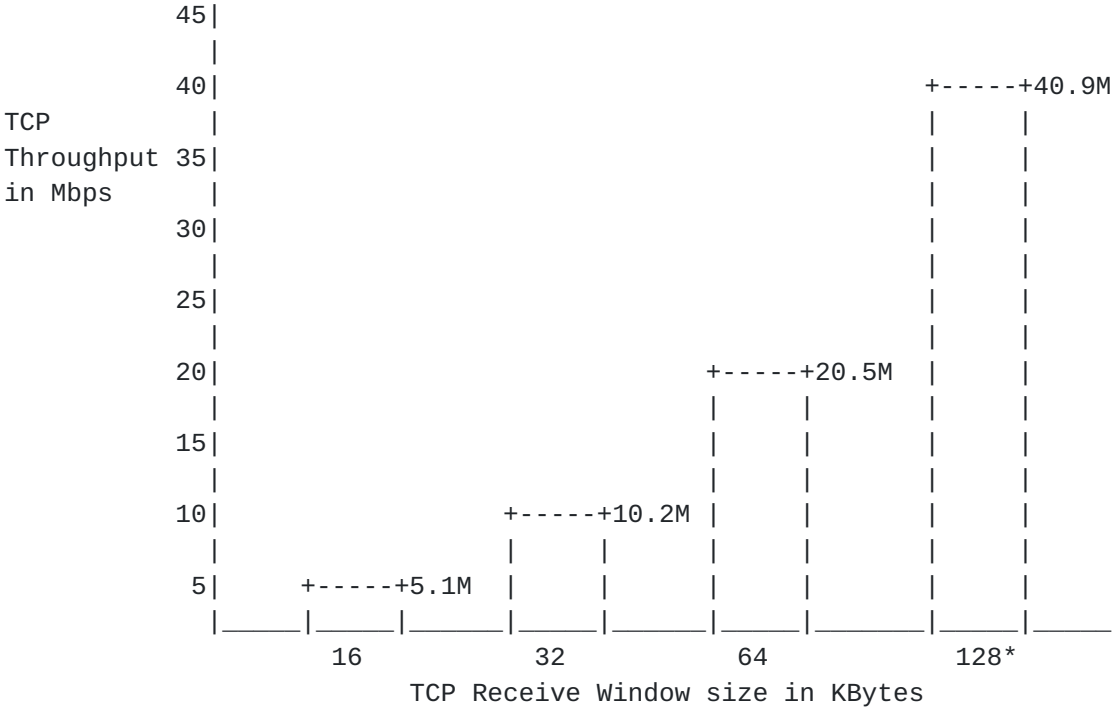
When the TCP Receive Window size exceeds the BDP (i.e. T3 link,
64 KBytes TCP Receive Window on a 10 ms RTT path), the maximum frames
per second limit of 3664 is reached and the calculation formula is:

```
   TCP Throughput = Max FPS X MSS X 8.
   TCP Throughput = 3664 FPS X 1460 Bytes X 8 bits.
   TCP Throughput = 42.8 Mbps
```

The following diagram compares achievable TCP throughputs on a T3 with Send Socket Buffer & TCP Receive Window sizes of 16KB vs. 64KB.



The following diagram shows the achievable TCP throughput on a 25ms T3 when Send Socket Buffer & TCP Receive Window sizes are increased.



* Note that 128KB requires [RFC1323] TCP Window scaling option.

Note that some TCP/IP stack implementations are using Receive Window Auto-Tuning and cannot be adjusted until the feature is disabled.

3.3.2 Metrics for TCP Throughput Tests

This framework focuses on a TCP throughput methodology and also provides several basic metrics to compare results of various throughput tests. It is recognized that the complexity and unpredictability of TCP makes it impossible to develop a complete set of metrics that accounts for the myriad of variables (i.e. RTT variation, loss conditions, TCP implementation, etc.). However, these basic metrics will facilitate TCP throughput comparisons under varying network conditions and between network traffic management techniques.

The first metric is the TCP Transfer Time, which is simply the measured time it takes to transfer a block of data across simultaneous TCP connections. This concept is useful when benchmarking traffic management techniques and where multiple TCP connections are required.

TCP Transfer time may also be used to provide a normalized ratio of the actual TCP Transfer Time versus the Ideal Transfer Time. This ratio is called the TCP Transfer Index and is defined as:

$$\frac{\text{Actual TCP Transfer Time}}{\text{Ideal TCP Transfer Time}}$$

The Ideal TCP Transfer time is derived from the network path bottleneck bandwidth and various Layer 1/2/3/4 overheads associated with the network path. Additionally, both the TCP Receive Window and the Send Socket Buffer sizes must be tuned to equal the bandwidth delay product (BDP) as described in section 3.3.1.

The following table illustrates the Ideal TCP Transfer time of a single TCP connection when its TCP Receive Window and Send Socket Buffer sizes are equal to the BDP.

Table 3.3.2: Link Speed, RTT, BDP, TCP Throughput, and Ideal TCP Transfer time for a 100 MB File

Link Speed (Mbps)	RTT (ms)	BDP (KBytes)	Maximum Achievable TCP Throughput(Mbps)	Ideal TCP Transfer time (seconds)
1.536	50	9.6	1.4	571
44.21	25	138.2	42.8	18
100	2	25.0	94.9	9
1,000	1	125.0	949.2	1
10,000	0.05	62.5	9,492	0.1

Transfer times are rounded for simplicity.

For a 100MB file(100 x 8 = 800 Mbits), the Ideal TCP Transfer Time is derived as follows:

$$\text{Ideal TCP Transfer Time} = \frac{800 \text{ Mbits}}{\text{Maximum Achievable TCP Throughput}}$$

The maximum achievable layer 2 throughput on T1 and T3 Interfaces is based on the maximum frames per second (FPS) permitted by the actual layer 1 speed when the MTU is 1500 Bytes.

The maximum FPS for a T1 is 127 and the calculation formula is:
FPS = T1 Link Speed / ((MTU + PPP + Flags + CRC16) X 8)
FPS = (1.536M /((1500 Bytes + 4 Bytes + 2 Bytes + 2 Bytes) X 8)))
FPS = (1.536M / (1508 Bytes X 8))
FPS = 1.536 Mbps / 12064 bits
FPS = 127

The maximum FPS for a T3 is 3664 and the calculation formula is:
FPS = T3 Link Speed / ((MTU + PPP + Flags + CRC16) X 8)
FPS = (44.21M /((1500 Bytes + 4 Bytes + 2 Bytes + 2 Bytes) X 8)))
FPS = (44.21M / (1508 Bytes X 8))
FPS = 44.21 Mbps / 12064 bits
FPS = 3664

The 1508 equates to:

MTU + PPP + Flags + CRC16

Where MTU is 1500 Bytes, PPP is 4 Bytes, Flags are 2 Bytes and CRC16 is 2 Bytes.

Then, to obtain the Maximum Achievable TCP Throughput (layer 4), we simply use: MSS in Bytes X 8 bits X max FPS.
For a T3, the maximum TCP Throughput = 1460 Bytes X 8 bits X 3664 FPS
Maximum TCP Throughput = 11680 bits X 3664 FPS
Maximum TCP Throughput = 42.8 Mbps.

The maximum achievable layer 2 throughput on Ethernet Interfaces is based on the maximum frames per second permitted by the IEEE802.3 standard when the MTU is 1500 Bytes.

The maximum FPS for 100M Ethernet is 8127 and the calculation is:
FPS = (100Mbps /(1538 Bytes X 8 bits))

The maximum FPS for GigE is 81274 and the calculation formula is:
FPS = (1Gbps /(1538 Bytes X 8 bits))

The maximum FPS for 10GigE is 812743 and the calculation formula is:
FPS = (10Gbps /(1538 Bytes X 8 bits))

The 1538 equates to:

MTU + Eth + CRC32 + IFG + Preamble + SFD

Where MTU is 1500 Bytes, Ethernet is 14 Bytes, CRC32 is 4 Bytes, IFG is 12 Bytes, Preamble is 7 Bytes and SFD is 1 Byte.

Note that better results could be obtained with jumbo frames on GigE and 10 GigE.

Then, to obtain the Maximum Achievable TCP Throughput (layer 4), we simply use: MSS in Bytes X 8 bits X max FPS.
For a 100M, the maximum TCP Throughput = 1460 B X 8 bits X 8127 FPS
Maximum TCP Throughput = 11680 bits X 8127 FPS
Maximum TCP Throughput = 94.9 Mbps.

To illustrate the TCP Transfer Time Index, an example would be the bulk transfer of 100 MB over 5 simultaneous TCP connections (each connection uploading 100 MB). In this example, the Ethernet service provides a Committed Access Rate (CAR) of 500 Mbit/s. Each connection may achieve different throughputs during a test and the overall throughput rate is not always easy to determine (especially as the number of connections increases).

The ideal TCP Transfer Time would be ~8 seconds, but in this example, the actual TCP Transfer Time was 12 seconds. The TCP Transfer Index would then be 12/8 = 1.5, which indicates that the transfer across all connections took 1.5 times longer than the ideal.

The second metric is TCP Efficiency, which is the percentage of Bytes that were not retransmitted and is defined as:

$$\frac{\text{Transmitted Bytes} - \text{Retransmitted Bytes}}{\text{Transmitted Bytes}} \times 100$$

Transmitted Bytes are the total number of TCP payload Bytes to be transmitted which includes the original and retransmitted Bytes. This metric provides a comparative measure between various QoS mechanisms like traffic management or congestion avoidance. Various TCP implementations like Reno, Vegas, etc. could also be compared.

As an example, if 100,000 Bytes were sent and 2,000 had to be retransmitted, the TCP Efficiency should be calculated as:

$$\frac{102{,}000 - 2{,}000}{102{,}000} \times 100 = 98.03\%$$

Note that the retransmitted Bytes may have occurred more than once, and these multiple retransmissions are added to the Retransmitted Bytes count (and the Transmitted Bytes count).

The third metric is the Buffer Delay Percentage, which represents the increase in RTT during a TCP throughput test with respect to inherent or baseline network RTT. The baseline RTT is the round-trip time inherent to the network path under non-congested conditions. (See 3.2.1 for details concerning the baseline RTT measurements).

The Buffer Delay Percentage is defined as:

$$\frac{\text{Average RTT during Transfer} - \text{Baseline RTT}}{\text{Baseline RTT}} \times 100$$

As an example, the baseline RTT for the network path is 25 msec. During the course of a TCP transfer, the average RTT across the entire transfer increased to 32 msec. In this example, the Buffer Delay Percentage would be calculated as:

$$\frac{32 - 25}{25} \times 100 = 28\%$$

Note that the TCP Transfer Time, TCP Efficiency, and Buffer Delay Percentage MUST be measured during each throughput test. Poor TCP Transfer Time Indexes (TCP Transfer Time greater than Ideal TCP Transfer Times) may be diagnosed by correlating with sub-optimal TCP Efficiency and/or Buffer Delay Percentage metrics.

3.3.3 Conducting the TCP Throughput Tests

Several TCP tools are currently used in the network world and one of the most common is "iperf". With this tool, hosts are installed at each end of the network path; one acts as client and the other as a server. The Send Socket Buffer and the TCP Receive Window sizes of both client and server can be manually set. The achieved throughput can then be measured, either uni-directionally or bi-directionally. For higher BDP situations in lossy networks (long fat networks or satellite links, etc.), TCP options such as Selective Acknowledgment SHOULD be considered and become part of the window size / throughput characterization.

Note that some TCP/IP stack implementations are using Receive Window Auto-Tuning and cannot be adjusted until this feature is disabled.

Host hardware performance must be well understood before conducting the tests described in the following sections. A dedicated communications test instrument will generally be required, especially for line rates of GigE and 10 GigE. A compliant TCP TTD SHOULD provide a warning message when the expected test throughput will exceed 10% of the network bandwidth capacity. If the throughput test

is expected to exceed 10% of the provider bandwidth, then the test should be coordinated with the network provider. This does not include the customer premise bandwidth, the 10% refers directly to the provider's bandwidth (Provider Edge to Provider router).

The TCP throughput test should be run over a long enough duration to properly exercise network buffers (greater than 30 seconds) and also characterize performance at different time periods of the day.

3.3.4 Single vs. Multiple TCP Connection Testing

The decision whether to conduct single or multiple TCP connection tests depends upon the size of the BDP in relation to the configured TCP Receive Window sizes configured in the end-user environment. For example, if the BDP for a long fat network turns out to be 2MB, then it is probably more realistic to test this network path with multiple connections. Assuming typical host computer TCP Receive Window Sizes of 64 KB, using 32 TCP connections would realistically test this path.

The following table is provided to illustrate the relationship between the TCP Receive Window size and the number of TCP connections required to utilize the available capacity of a given BDP. For this example, the network bandwidth is 500 Mbps and the RTT is 5 ms, then the BDP equates to 312.5 KBytes.

TCP Window	Number of TCP Connections to fill available bandwidth
16KB	20
32KB	10
64KB	5
128KB	3

Note that some TCP/IP stack implementations are using Receive Window Auto-Tuning and cannot be adjusted until this feature is disabled.

The TCP Transfer Time metric is useful for conducting multiple connection tests. Each connection should be configured to transfer payloads of the same size (i.e. 100 MB), and the TCP Transfer time should provide a simple metric to verify the actual versus expected results.

Note that the TCP transfer time is the time for all connections to complete the transfer of the configured payload size. From the previous table, the 64KB window is considered. Each of the 5 TCP connections would be configured to transfer 100MB, and each one should obtain a maximum of 100 Mb/sec. So for this example, the 100MB payload should be transferred across the connections in approximately 8 seconds (which would be the ideal TCP transfer time under these conditions).

Additionally, the TCP Efficiency metric MUST be computed for each connection tested as defined in section 3.3.2.

3.3.5 Interpretation of the TCP Throughput Results

At the end of this step, the user will document the theoretical BDP and a set of Window size experiments with measured TCP throughput for each TCP window size. For cases where the sustained TCP throughput does not equal the ideal value, some possible causes are:

- Network congestion causing packet loss which MAY be inferred from a poor TCP Efficiency % (higher TCP Efficiency % = less packet loss)
- Network congestion causing an increase in RTT which MAY be inferred from the Buffer Delay Percentage (i.e., 0% = no increase in RTT over baseline)
- Intermediate network devices which actively regenerate the TCP connection and can alter TCP Receive Window size, MSS, etc.
- Rate limiting (policing). More details on traffic management tests follows in section 3.4

3.4. Traffic Management Tests

In most cases, the network connection between two geographic locations (branch offices, etc.) is lower than the network connection to host computers. An example would be LAN connectivity of GigE and WAN connectivity of 100 Mbps. The WAN connectivity may be physically 100 Mbps or logically 100 Mbps (over a GigE WAN connection). In the later case, rate limiting is used to provide the WAN bandwidth per the SLA.

Traffic management techniques are employed to provide various forms of QoS, the more common include:

- Traffic Shaping
- Priority queuing
- Random Early Discard (RED)

Configuring the end-to-end network with these various traffic management mechanisms is a complex under-taking. For traffic shaping and RED techniques, the end goal is to provide better performance to bursty traffic such as TCP,(RED is specifically intended for TCP).

This section of the methodology provides guidelines to test traffic shaping and RED implementations. As in section 3.3, host hardware performance must be well understood before conducting the traffic shaping and RED tests. Dedicated communications test instrument will generally be REQUIRED for line rates of GigE and 10 GigE. If the throughput test is expected to exceed 10% of the provider bandwidth, then the test should be coordinated with the network provider. This does not include the customer premises bandwidth, the 10% refers to the provider's bandwidth (Provider Edge to Provider router). Note

that GigE and 10 GigE interfaces might benefit from hold-queue adjustments in order to prevent the saw-tooth TCP traffic pattern.

3.4.1 Traffic Shaping Tests

For services where the available bandwidth is rate limited, two (2) techniques can be used: traffic policing or traffic shaping.

Simply stated, traffic policing marks and/or drops packets which exceed the SLA bandwidth (in most cases, excess traffic is dropped). Traffic shaping employs the use of queues to smooth the bursty traffic and then send out within the SLA bandwidth limit (without dropping packets unless the traffic shaping queue is exhausted).

Traffic shaping is generally configured for TCP data services and can provide improved TCP performance since the retransmissions are reduced, which in turn optimizes TCP throughput for the available bandwidth. Through this section, the rate-limited bandwidth shall be referred to as the "bottleneck bandwidth".

The ability to detect proper traffic shaping is more easily diagnosed when conducting a multiple TCP connections test. Proper shaping will provide a fair distribution of the available bottleneck bandwidth, while traffic policing will not.

The traffic shaping tests are built upon the concepts of multiple connections testing as defined in section 3.3.3. Calculating the BDP for the bottleneck bandwidth is first required before selecting the number of connections, the Send Socket Buffer and TCP Receive Window sizes per connection.

Similar to the example in section 3.3, a typical test scenario might be: GigE LAN with a 100Mbps bottleneck bandwidth (rate limited logical interface), and 5 msec RTT. This would require five (5) TCP connections of 64 KB Send Socket Buffer and TCP Receive Window sizes to evenly fill the bottleneck bandwidth (~100 Mbps per connection).

The traffic shaping test should be run over a long enough duration to properly exercise network buffers (greater than 30 seconds) and also characterize performance during different time periods of the day. The throughput of each connection MUST be logged during the entire test, along with the TCP Transfer Time, TCP Efficiency, and Buffer Delay Percentage.

3.4.1.1 Interpretation of Traffic Shaping Test Results

By plotting the throughput achieved by each TCP connection, the fair sharing of the bandwidth is generally very obvious when traffic shaping is properly configured for the bottleneck interface. For the previous example of 5 connections sharing 500 Mbps, each connection would consume ~100 Mbps with a smooth variation.

If traffic policing was present on the bottleneck interface, the

bandwidth sharing may not be fair and the resulting throughput plot may reveal "spikey" throughput consumption of the competing TCP connections (due to the TCP retransmissions).

3.4.2 RED Tests

Random Early Discard techniques are specifically targeted to provide congestion avoidance for TCP traffic. Before the network element queue "fills" and enters the tail drop state, RED drops packets at configurable queue depth thresholds. This action causes TCP connections to back-off which helps to prevent tail drop, which in turn helps to prevent global TCP synchronization.

Again, rate limited interfaces may benefit greatly from RED based techniques. Without RED, TCP may not be able to achieve the full bottleneck bandwidth. With RED enabled, TCP congestion avoidance throttles the connections on the higher speed interface (i.e. LAN) and can help achieve the full bottleneck bandwidth. The burstiness of TCP traffic is a key factor in the overall effectiveness of RED techniques; steady state bulk transfer flows will generally not benefit from RED. With bulk transfer flows, network device queues gracefully throttle the effective throughput rates due to increased delays.

The ability to detect proper RED configuration is more easily diagnosed when conducting a multiple TCP connections test. Multiple TCP connections provide the bursty sources that emulate the real-world conditions for which RED was intended.

The RED tests also builds upon the concepts of multiple connections testing as defined in section 3.3.3. Calculating the BDP for the bottleneck bandwidth is first required before selecting the number of connections, the Send Socket Buffer size and the TCP Receive Window size per connection.

For RED testing, the desired effect is to cause the TCP connections to burst beyond the bottleneck bandwidth so that queue drops will occur. Using the same example from section 3.4.1 (traffic shaping), the 500 Mbps bottleneck bandwidth requires 5 TCP connections (with window size of 64KB) to fill the capacity. Some experimentation is required, but it is recommended to start with double the number of connections to stress the network element buffers / queues (10 connections for this example).

The TCP TTD must be configured to generate these connections as shorter (bursty) flows versus bulk transfer type flows. These TCP bursts should stress queue sizes in the 512KB range. Again experimentation will be required; the proper number of TCP connections, the Send Socket Buffer and TCP Receive Window sizes will be dictated by the size of the network element queue.

3.4.2.1 Interpretation of RED Results

The default queuing technique for most network devices is FIFO based. Without RED, the FIFO based queue may cause excessive loss to all of the TCP connections and in the worst case global TCP synchronization.

By plotting the aggregate throughput achieved on the bottleneck interface, proper RED operation may be determined if the bottleneck bandwidth is fully utilized. For the previous example of 10 connections (window = 64 KB) sharing 500 Mbps, each connection should consume ~50 Mbps. If RED was not properly enabled on the interface, then the TCP connections will retransmit at a higher rate and the net effect is that the bottleneck bandwidth is not fully utilized.

Another means to study non-RED versus RED implementation is to use the TCP Transfer Time metric for all of the connections. In this example, a 100 MB payload transfer should take ideally 16 seconds across all 10 connections (with RED enabled). With RED not enabled, the throughput across the bottleneck bandwidth may be greatly reduced (generally 10-20%) and the actual TCP Transfer time may be proportionally longer then the Ideal TCP Transfer time.

Additionally, non-RED implementations may exhibit a lower TCP Transfer Efficiency.

4. Security Considerations

The security considerations that apply to any active measurement of live networks are relevant here as well. See [RFC4656] and [RFC5357].

5. IANA Considerations

This document does not REQUIRE an IANA registration for ports dedicated to the TCP testing described in this document.

6. Acknowledgments

Thanks to Lars Eggert, Al Morton, Matt Mathis, Matt Zekauskas, Yaakov Stein, and Loki Jorgenson for many good comments and for pointing us to great sources of information pertaining to past works in the TCP capacity area.

7. References

7.1 Normative References

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

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[RFC2544] Bradner, S., McQuaid, J., "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, June 1999

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., Babiarz, J., "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008

[RFC4821] Mathis, M., Heffner, J., "Packetization Layer Path MTU Discovery", RFC 4821, June 2007

draft-ietf-ippm-btc-cap-00.txt Allman, M., "A Bulk Transfer Capacity Methodology for Cooperating Hosts", August 2001

[RFC2681] Almes G., Kalidindi S., Zekauskas, M., "A Round-trip Delay Metric for IPPM", RFC 2681, September, 1999

[RFC4898] Mathis, M., Heffner, J., Raghunarayan, R., "TCP Extended Statistics MIB", May 2007

[RFC5136] Chimento P., Ishac, J., "Defining Network Capacity", February 2008

[RFC1323] Jacobson, V., Braden, R., Borman D., "TCP Extensions for High Performance", May 1992

7.2. Informative References

Authors' Addresses

Barry Constantine
JDSU, Test and Measurement Division
One Milesone Center Court
Germantown, MD 20876-7100
USA

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

Gilles Forget
Independent Consultant to Bell Canada.
308, rue de Monaco, St-Eustache
Qc. CANADA, Postal Code : J7P-4T5

Phone: (514) 895-8212
gilles.forget@sympatico.ca

Rudiger Geib
Heinrich-Hertz-Strasse (Number: 3-7)
Darmstadt, Germany, 64295

Phone: +49 6151 6282747
Ruediger.Geib@telekom.de

Reinhard Schrage
Schrage Consulting

Phone: +49 (0) 5137 909540
reinhard@schrageconsult.com