Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 31, 2011

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

January 31, 2011

Framework for TCP Throughput Testing
draft-ietf-ippm-tcp-throughput-tm-11.txt

Abstract

This framework describes a practical methodology for measuring end-to-end TCP throughput in a managed IP network. The goal is to provide a better indication in regards to user experience. In this framework, TCP and IP parameters are specified and should be configured as recommended.

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 July 31, 2011.

Table of Contents

1. Introduction

The SLA (Service Level Agreement) provided to business class customers is generally based upon Layer 2/3 criteria such as : Guaranteed bandwidth, maximum network latency, maximum packet loss percentage and maximum delay variation (i.e. maximum jitter). Network providers are coming to the realization that Layer 2/3 testing is not enough to adequately ensure end-user's satisfaction. In addition to Layer 2/3 performance, measuring TCP throughput provides more meaningful results with respect to user experience.

Additionally, business class customers seek to conduct repeatable TCP throughput tests between locations. Since these organizations rely on the networks of the providers, a common test methodology with predefined metrics would 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 organizations are provided an SLA from the network provider. Because of the SLA, the expectation is that the TCP Throughput should achieve the guaranteed bandwidth. End-users with "best effort" access could use this methodology, but this framework and its metrics are intended to be used in a predictable managed IP network. No end-to-end performance can be guaranteed when only the access portion is being provisioned to a specific bandwidth capacity.

The intent behind this document is to define a methodology for testing sustained TCP layer performance. In this document, the achievable TCP Throughput is that amount of data per unit time that TCP transports when in the TCP Equilibrium state. (See section 2.1 for TCP Equilibrium definition). Throughout this document, maximum achievable throughput refers to the theoretical achievable throughput when TCP is in the Equilibrium state.

TCP is connection oriented and at the transmitting side it uses a congestion window, (TCP CWND). At the receiving end, TCP uses a receive window, (TCP RWND) to inform the transmitting end on how many Bytes it is capable to accept at a given time.

Derived from Round Trip Time (RTT) and network path bandwidth, the bandwidth delay product (BDP) determines the Send and Received Socket buffers sizes required to achieve the maximum TCP throughput. Then, with the help of slow start and congestion avoidance algorithms, a TCP CWND is calculated based on the IP network path loss rate. Finally, the minimum value between the calculated TCP CWND and the TCP RWND advertised by the opposite end will determine how many Bytes can actually be sent by the transmitting side at a given time.

Both TCP Window sizes (RWND and CWND) may vary during any given TCP session, although up to bandwidth limits, larger RWND and larger CWND will achieve higher throughputs by permitting more in-flight Bytes.

At both ends of the TCP connection and for each socket, there are default buffer sizes. There are also kernel enforced maximum buffer sizes. These buffer sizes can be adjusted at both ends (transmitting and receiving). Some TCP/IP stack implementations use Receive Window Auto-Tuning, although in order to obtain the maximum throughput it is critical to use large enough TCP Send and Receive Socket Buffer sizes. In fact, they should be equal to or greater than BDP.

Many variables are involved in TCP throughput performance, but this methodology focuses on:
- BB (Bottleneck Bandwidth)
- RTT (Round Trip Time)
- Send and Receive Socket Buffers
- Minimum TCP RWND
- Path MTU (Maximum Transmission Unit)
- Path MSS (Maximum Segment Size)

This methodology proposes TCP testing that should be performed in addition to traditional Layer 2/3 type tests. In fact, Layer 2/3 tests are required to verify the integrity of the network before conducting TCP tests. Examples include iperf (UDP mode) and manual packet layer test techniques where packet throughput, loss, and delay measurements are conducted. When available, standardized testing similar to [RFC2544] but adapted for use in operational networks may be used.

Note: RFC 2544 was never meant to be used outside a lab environment.

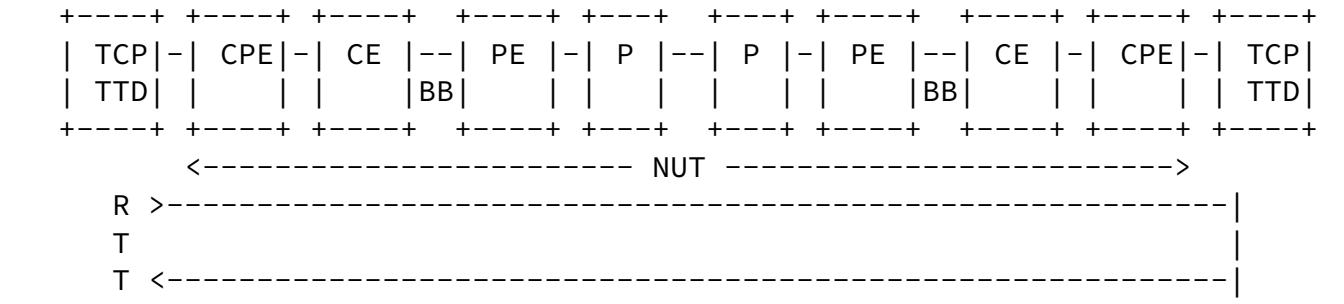
Sections 2 and 3 of this document provide a general overview of the proposed methodology.

1.1 Terminology

The common definitions used in this 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.

Figure 1.1 Devices, Links and Paths



Note that the NUT may be built with of a variety of devices including but not limited to, load balancers, proxy servers or WAN acceleration appliances. The detailed topology of the NUT should be well known 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 diagram is very general and is only there to illustrate typical 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 during the initial slow start phase.

- This methodology is not intended to definitively benchmark TCP implementations of one OS to another, although some users may find 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 insights to 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, the 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 IP network to carry end-user TCP applications.

Specific goals are to :

- Provide a practical test approach that specifies tunable parameters

(such as MSS (Maximum Segment Size) and Socket Buffer sizes) and how these affect the outcome of TCP performances over an IP network. See section 3.3.3.

- Provide specific test conditions like link speed, RTT, MSS, Socket Buffer sizes and achievable TCP throughput when TCP is in the Equilibrium state. 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.

- 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, 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 detailed diagnosis on these issues. See section 3.3.5.

2.1 TCP Equilibrium

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

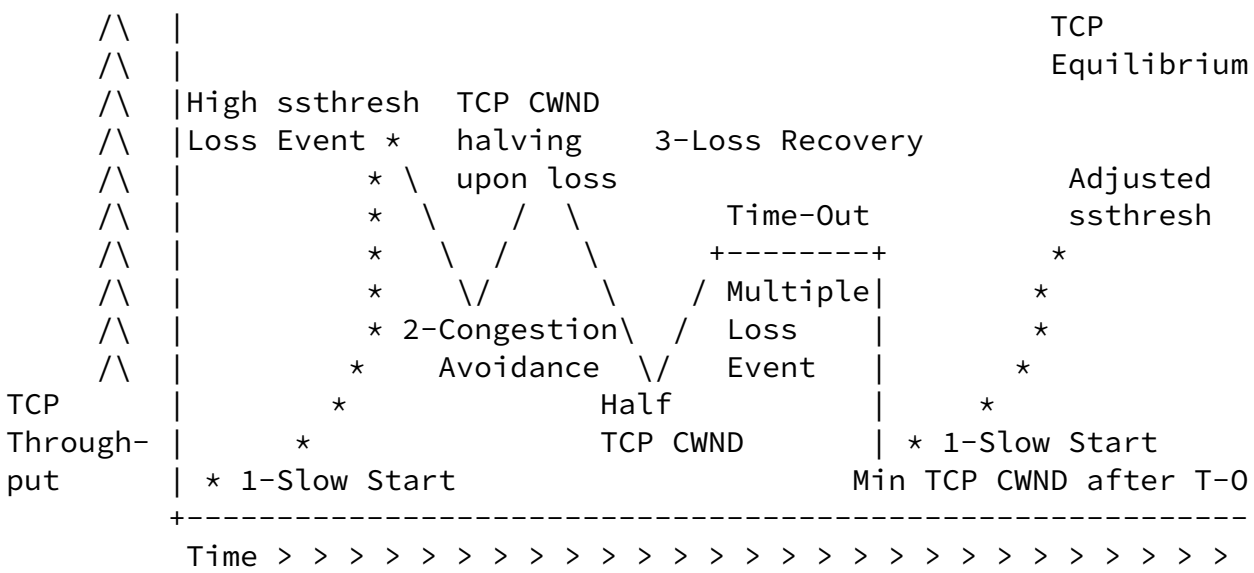
1 - The Slow Start phase, which occurs at the beginning of a TCP transmission or after a retransmission time out.

2 - The Congestion Avoidance phase, during which TCP ramps up to establish the maximum achievable throughput. It is important to note that 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 the TCP implementation. If a Time-Out occurs, TCP transitions back to the Slow Start phase.

The following diagram depicts these 3 phases.

Figure 2.1 TCP CWND Phases



Note : ssthresh = Slow Start threshold.

A well tuned and managed IP network with appropriate TCP adjustments in the IP hosts and applications should perform very close to the BB (Bottleneck Bandwidth) when TCP is in the Equilibrium state.

This TCP methodology provides guidelines to measure the maximum achievable TCP throughput when TCP is in the Equilibrium state. 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 RWND Size. TCP test programs such as iperf, ttcp, etc. allows 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 RWND 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 Layer 2/3 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 achieve optimal TCP throughputs in regards with the available bandwidth.

TCP Throughput testing may require cooperation between the end-user customer and the network provider. As an example, in an MPLS (Multi-Protocol Label Switching) network 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 TCP RWND and Send Socket Buffer Sizes that SHOULD be used during subsequent test steps. These measurements refers to [RFC2681] and [RFC4898] in order to measure RTD and 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 performances.

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 and that Active Queue Management implementations are used.

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, it must be capable of emulating both a client and a 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 Socket Buffers MUST be large enough to fill the BDP.

- 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. 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 independently 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, (20 for IP + 20 for TCP) then the MSS would be 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 advertised MSS 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 it would be

set to a MSS of 1460 in order to produce 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 none of the test packets are ACK'ed, congestive network may be the cause and the test probe is not conclusive. Re-testing at another time 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 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 (i.e. non-congested inherent delay) and Bottleneck Bandwidth of the end-to-end network to be tested. These measurements are used to calculate the BDP and to provide estimates of the TCP RWND 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 and 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 in order to obtain a reliable figure of the inherent network latency. Otherwise, additional delay caused by network buffering can occur. Also, when sampling RTT values over a given test interval, the minimum measured value SHOULD be used as the baseline RTT. This will most closely estimate the real inherent RTT. This value is also used to determine the Buffer Delay Percentage metric 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 measurement precision (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. Also, at the senders side, Ostermann's LINUX TCPTRACE utility with -l -r arguments can be used to extract the RTT results directly from the packet captures.

- 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 or segregated into different buffer queues. ICMP might not work if QoS (Quality of Service) reclassification is done at any hop. ICMP is not as reliable and accurate as in-band measurements.

3.2.2 Techniques to Measure end-to-end Bandwidth

Before any TCP Throughput test can be conducted, bandwidth measurement tests MUST be run with stateless IP streams (i.e. not stateful TCP) in order to determine the available path bandwidth. These measurements SHOULD be conducted in both directions, especially in asymmetrical access networks (e.g. ADSL access). These tests should obviously be performed at various intervals throughout a business day or even across a week. Ideally, the bandwidth tests should produce logged outputs of the achieved bandwidths across the complete test duration.

There are many well established techniques available to provide estimated measures of bandwidth over a network. It is a common practice for network providers to conduct Layer 2/3 bandwidth capacity tests using [RFC2544], although it is understood that [RFC2544] was never meant to be used outside a lab environment. Ideally, these bandwidth measurements SHOULD use network capacity techniques as defined in [RFC5136].

3.3. TCP Throughput Tests

This methodology specifically defines TCP throughput techniques to verify maximum achievable TCP performance in a managed business class IP network, as defined in section 2.1. This document defines a method to conduct these maximum achievable TCP throughput tests as well as guidelines on 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 in order to measure network performance against expectations. The number of trials and the type of testing (i.e. 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 and Receive Socket Buffer sizes (i.e. 256KB) is to be measured. It would be advisable to test 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, the TCP Transfer Time, TCP Efficiency, and Buffer Delay Percentage metrics MUST be measured in each direction. These metrics are defined in 3.3.2.

3.3.1 Calculate minimum required TCP RWND Size

The minimum required TCP RWND 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 minimum required TCP

RWND 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.

Minimum required TCP RWND = BDP / 8

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

Note that separate calculations are required on asymmetrical paths. An asymmetrical 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 minimum required TCP RWND Sizes.

Table 3.3.1: Link Speed, RTT, calculated BDP & minimum TCP RWND

Link Speed* (Mbps)	RTT (ms)	BDP (bits)	Minimum required TCP RWND (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 (BB) 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 minimum required TCP RWND.
If a smaller TCP RWND Size is used, then the TCP Throughput can not be optimal. To calculate the TCP Throughput, the following formula is used: TCP Throughput = TCP RWND X 8 / RTT

An example could be a 100 Mbps IP path with 5 ms RTT and a TCP RWND 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:

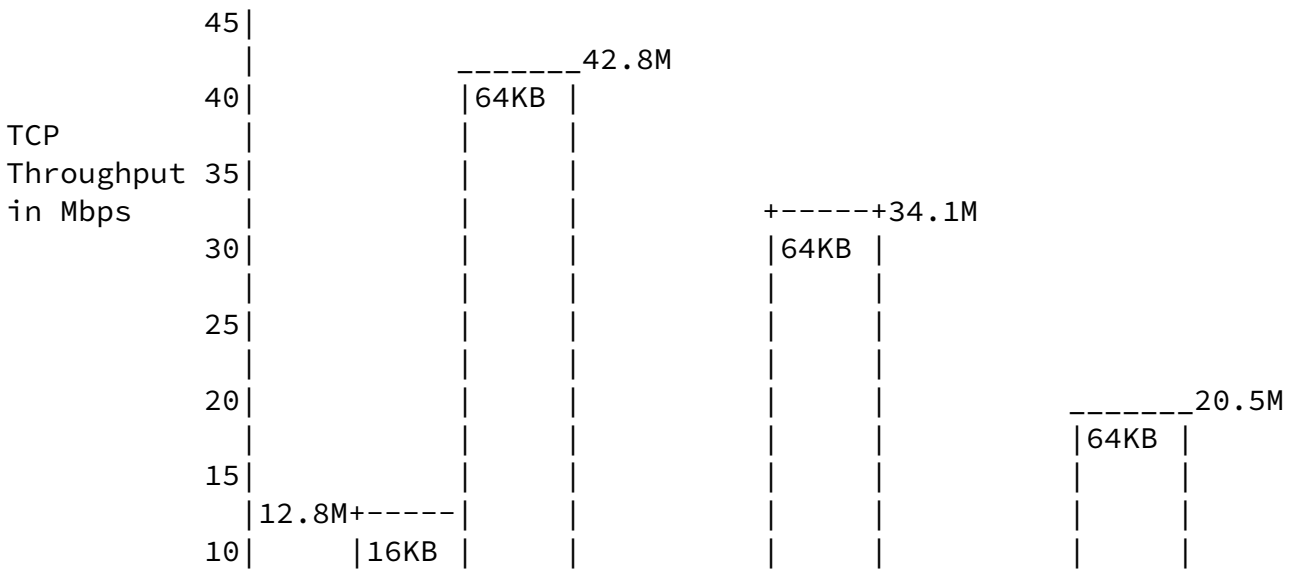
TCP Throughput = 16 KBytes X 8 bits / 10 ms.
TCP Throughput = 128,000 bits / 0.01 sec.
TCP Throughput = 12.8 Mbps.

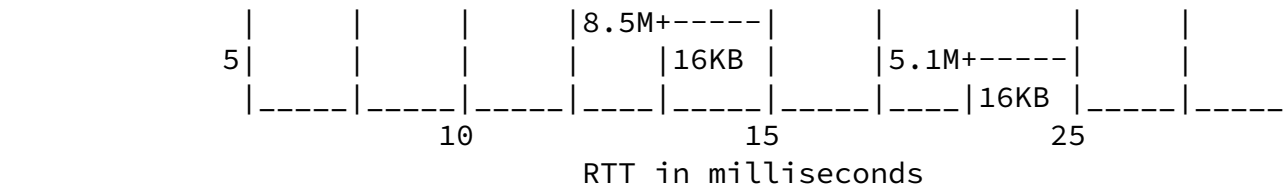
When the TCP RWND Size exceeds the BDP (T3 link and 64 KBytes TCP RWND on a 10 ms RTT path), the maximum frames per second limit of 3664 is reached and then the formula is:

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 RWND Sizes of 16KB vs. 64KB.

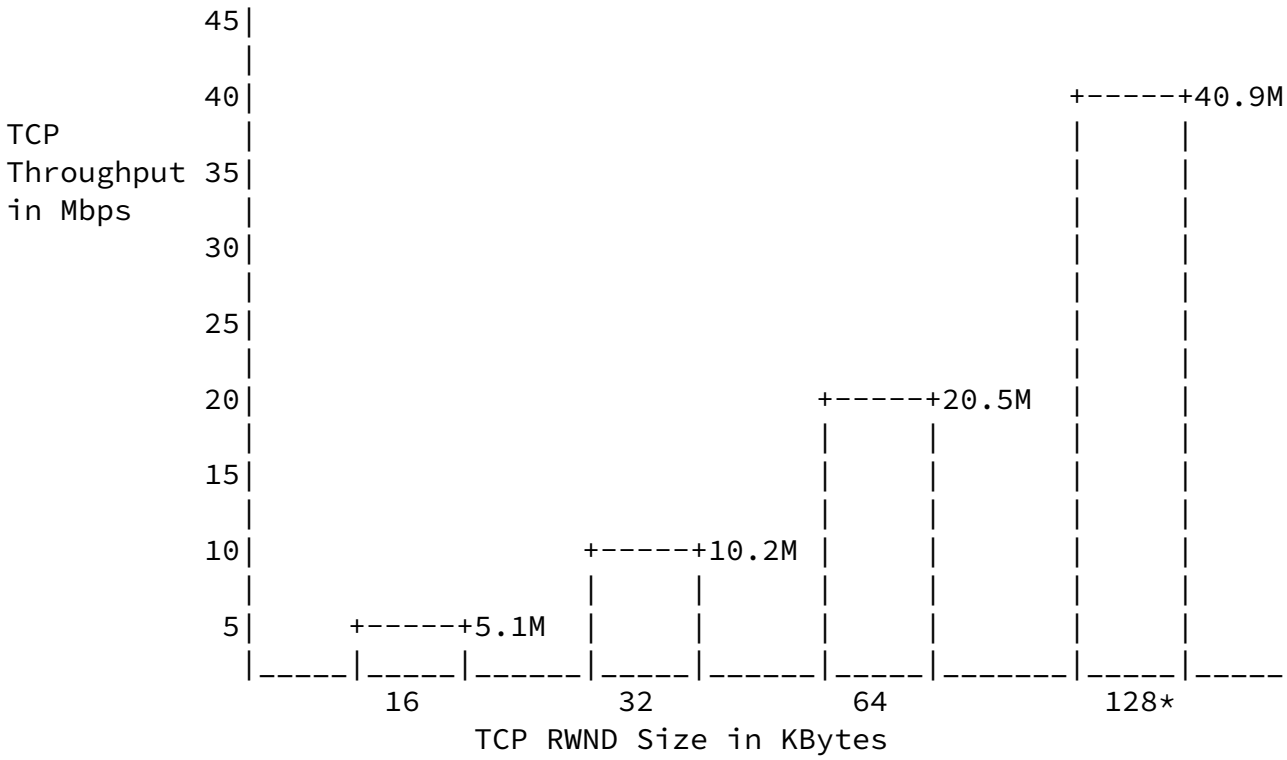
Figure 3.3.1a TCP Throughputs on a T3 at different RTTs





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

Figure 3.3.1b TCP Throughputs on a T3 with different TCP RWND



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

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 between 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 required to transfer a block of data across simultaneous TCP connections. This concept is useful when benchmarking traffic management techniques and when 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 Layer 1/2/3/4 overheads associated with the network path. Additionally, both the TCP RWND and the Send Socket Buffer Sizes must be tuned to equal or exceed 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 RWND and Send Socket Buffer Sizes equals or exceeds 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 with an MTU of 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 the MTU is 1500 Bytes, PPP is 4 Bytes, the 2 Flags are 1 Byte each and the 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
(IFG = Inter-Frame Gap and SFD = Start of Frame Delimiter)
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 transferring 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 Bytes to be transmitted including the original and the retransmitted Bytes. This metric provides comparative results between various traffic management and

congestion avoidance mechanisms. Performance between different TCP implementations could also be compared. (e.g. Reno, Vegas, etc).

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, if so, then these multiple retransmissions are added to the Retransmitted Bytes and to the Transmitted Bytes counts.

The third metric is the Buffer Delay Percentage, which represents the increase in RTT during a TCP throughput test versus the inherent or baseline 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, consider a network path with a baseline RTT of 25 msec. During the course of a TCP transfer, the average RTT across the entire transfer increases to 32 msec. Then, 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 RWND 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.

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 (i.e. greater than 30 seconds) and should also characterize performance at different times of 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 TCP RWND 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 RWND Sizes of 64 KB (i.e. Windows XP), using 32 TCP connections would emulate a typical small office scenario.

The following table is provided to illustrate the relationship between the TCP RWND and the number of TCP connections required to fill 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.

Table 3.3.4 Number of TCP connections versus TCP RWND

TCP RWND	Number of TCP Connections to fill available bandwidth
16KB	20
32KB	10
64KB	5

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 provides 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 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 RWND Size, MSS, etc.
- Rate limiting (policing). More details on traffic management tests follows in section 3.4

3.3.6 High Performance Network Options

For cases where the network outperforms the client/server IP hosts some possible causes are:

- Maximum TCP Buffer space. All operating systems have a global mechanism to limit the quantity of system memory to be used by TCP connections. On some systems, each connection is subject to a memory limit that is applied to the total memory used for input data, output data and controls. On other systems, there are separate limits for input and output buffer spaces per connection. Client/server IP hosts might be configured with Maximum Buffer Space limits that are far too small for high performance networks.

- Socket Buffer Sizes. Most operating systems support separate per connection send and receive buffer limits that can be adjusted as long as they stay within the maximum memory limits. These socket buffers must be large enough to hold a full BDP of TCP Bytes plus some overhead. There are several methods that can be used to adjust socket buffer sizes, but TCP Auto-Tuning automatically adjusts these as needed to optimally balance TCP performance and memory usage. It is important to note that Auto-Tuning is enabled by default in LINUX since the kernel release 2.6.6 and in UNIX since FreeBSD 7.0. It is also enabled by default in Windows since Vista and in MAC since OS X version 10.5 (leopard). Over buffering can cause some applications to behave poorly, typically causing sluggish interactive response and risk running the system out of memory. Large default socket buffers have to be considered carefully on multi-user systems.

- TCP Window Scale Option, RFC1323. This option enables TCP to support large BDP paths. It provides a scale factor which is required for TCP to support window sizes larger than 64KB. Most systems automatically request WSCALE under some conditions, such as when the receive socket buffer is larger than 64KB or when the other end of the TCP connection requests it first. WSCALE can only be negotiated during the 3 way handshake. If either end fails to request WSCALE or requests an insufficient value, it cannot be renegotiated. Different systems use different algorithms to select WSCALE, but it is very important to have large enough buffer sizes. Note that under these constraints, a client application wishing to send data at high rates may need to set its own receive buffer to something larger than 64K Bytes before it opens the connection to ensure that the server properly negotiates WSCALE. A system administrator might have to explicitly enable RFC1323 extensions. Otherwise, the client/server IP host would not support TCP window sizes (BDP) larger than 64KB. Most of the time, performance gains will be obtained by enabling this option in Long

Fat Networks. (i.e., networks with large BDP, see Figure 3.3.1b).

- TCP Timestamps Option, RFC1323. This feature provides better measurements of the Round Trip Time and protects TCP from data corruption that might occur if packets are delivered so late that the sequence numbers wrap before they are delivered. Wrapped sequence numbers do not pose a serious risk below 100 Mbps, but the risk increases at higher data rates. Most of the time, performance gains will be obtained by enabling this option in Gigabit bandwidth networks.

- TCP Selective Acknowledgments Option (SACK), RFC2018. This allows a TCP receiver to inform the sender about exactly which data segment is missing and needs to be retransmitted. Without SACK, TCP has to estimate which data segment is missing, which works just fine if all losses are isolated (i.e. only one loss in any given round trip). Without SACK, TCP takes a very long time to recover after multiple and consecutive losses. SACK is now supported by most operating systems, but it may have to be explicitly enabled by the system administrator. In networks with unknown load and error patterns, TCP SACK will improve throughput performances. On the other hand, security appliances vendors might have implemented TCP randomization without considering TCP SACK and under such circumstances, SACK might need to be disabled in the client/server IP hosts until the vendor corrects the issue. Also, poorly implemented SACK algorithms might cause extreme CPU loads and might need to be disabled.

- Path MTU. The client/server IP host system must use the largest possible MTU for the path. This may require enabling Path MTU Discovery (RFC1191 & RFC4821). Since RFC1191 is flawed it is sometimes not enabled by default and may need to be explicitly enabled by the system administrator. RFC4821 describes a new, more robust algorithm for MTU discovery and ICMP black hole recovery.

- TOE (TCP Offload Engine). Some recent Network Interface Cards (NIC) are equipped with drivers that can do part or all of the TCP/IP protocol processing. TOE implementations require additional work (i.e. hardware-specific socket manipulation) to set up and tear down connections. Because TOE NICs configuration parameters are vendor

specific and not necessarily RFC-compliant, they are poorly integrated with UNIX & LINUX. Occasionally, TOE might need to be disabled in a server because its NIC does not have enough memory resources to buffer thousands of connections.

Note that both ends of a TCP connection must be properly tuned.

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 might be employed and the most common are:

- Traffic Policing and/or Shaping
- Priority queuing
- Active Queue Management (AQM)

Configuring the end-to-end network with these various traffic management mechanisms is a complex under-taking. For traffic shaping and AQM techniques, the end goal is to provide better performance to bursty traffic.

This section of the methodology provides guidelines to test traffic shaping and AQM implementations. As in section 3.3, host hardware performance must be well understood before conducting the traffic shaping and AQM 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. Throughout 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 RWND 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 RWND 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 (i.e. greater than 30 seconds) and should also characterize performance at different times of 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, we should see fair sharing of the bandwidth when traffic shaping is properly configured. For the previous example of 5 connections sharing 500 Mbps, each connection would consume ~100 Mbps with smooth variations.

If traffic shaping is not configured properly or if traffic policing is present on the bottleneck interface, the bandwidth sharing may not be fair. The resulting throughput plot may reveal "spikey" throughput consumption of the competing TCP connections (due to the high rate of TCP retransmissions).

3.4.2 AQM Tests

Active Queue Management techniques are specifically targeted to provide congestion avoidance to TCP traffic. As an example, before the network element queue "fills" and enters the tail drop state, an AQM implementation like RED (Random Early Discard) drops packets at pre-configurable queue depth thresholds. This action causes TCP connections to back-off which helps prevent tail drops and in turn helps avoid global TCP synchronization.

RED is just an example and other AQM implementations like WRED (Weighted Random Early Discard) or REM (Random Exponential Marking) or AREM (Adaptive Random Exponential Marking), just to name a few, could be used.

Again, rate limited interfaces may benefit greatly from AQM based techniques. With a default FIFO queue, bloated buffering is increasingly a common encounter and has dire effects on TCP connections. However, the main effect is the delayed congestion feedback (poor TCP control loop response) and enormous queuing delays on all other traffic flows.

In a FIFO based queue, the TCP traffic may not be able to achieve the full throughput available on the Bottleneck Bandwidth link. While with an AQM implementation, TCP congestion avoidance would throttle the connections on the higher speed interface (i.e. LAN) and could help achieve the full throughput (up to the Bottleneck Bandwidth). The bursty nature of TCP traffic is a key factor in the overall effectiveness of AQM techniques; steady state bulk transfer flows will generally not benefit from AQM because with bulk transfer flows, network device queues gracefully throttle the effective throughput rates due to increased delays.

The ability to detect proper AQM 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 AQM implementations are intended.

AQM testing 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 RWND Size per connection.

For AQM 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 in order 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 RWND Sizes will be dictated by the size of the network element queue.

3.4.2.1 Interpretation of AQM Results

The default queuing technique for most network devices is FIFO based. Under heavy traffic conditions, FIFO based queue management may cause enormous queuing delays plus delayed congestion feedback to all TCP applications. This can cause excessive loss on all of the TCP connections and in the worst cases, global TCP synchronization.

AQM implementation can be detected by plotting individual and aggregate throughput results achieved by multiple TCP connections on the bottleneck interface. Proper AQM operation may be determined if the TCP throughput is fully utilized (up to the Bottleneck Bandwidth) and fairly shared between TCP connections. For the previous example of 10 connections (window = 64 KB) sharing 500 Mbps, each connection should consume ~50 Mbps. If AQM was not properly enabled on the interface, then the TCP connections would retransmit at higher rates and the net effect is that the Bottleneck Bandwidth is not fully utilized.

Another means to study non-AQM versus AQM implementations is to use the Buffer Delay Percent metric for all of the connections. The Buffer Delay Percentage should be significantly lower in AQM implementations versus default FIFO queuing.

Additionally, non-AQM 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