Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 12, 2011

B. Constantine
JDSU
G. Forget
Bell Canada (Ext. Consultant)
L. Jorgenson
nooCore
Reinhard Schrage
Schrage Consulting

August 12, 2010

TCP Throughput Testing Methodology
draft-ietf-ippm-tcp-throughput-tm-05.txt

Abstract

This memo describes a methodology for measuring sustained TCP throughput performance in an end-to-end managed network environment. This memo is intended to provide a practical approach to help users validate the TCP layer performance of a managed network, which should provide a better indication of end-user application level experience. In the methodology, various TCP and network parameters are identified that should be tested as part of the network verification at the TCP layer.

Status of this Memo

This Internet-Draft will expire on February 12, 2011.

Copyright Notice

Table of Contents

1. Introduction

Testing an operational network prior to customer activation is referred to as "turn-up" testing and the SLA is generally Layer 2/3 packet throughput, delay, loss and jitter.

Network providers are coming to the realization that Layer 2/3 testing and TCP layer testing are required to more adequately ensure end-user satisfaction. Therefore, the network provider community desires to measure network throughput performance at the TCP layer. Measuring TCP throughput provides a meaningful measure with respect to the end user's application SLA (and ultimately reach some level of TCP testing interoperability which does not exist today).

Additionally, end-users (business enterprises) seek to conduct repeatable TCP throughput tests between enterprise locations. Since these enterprises rely on the networks of the providers, a common test methodology (and metrics) would be equally beneficial to both parties.

So the intent behind this TCP throughput draft is to define a methodology for testing sustained TCP layer performance. In this document, sustained TCP throughput is that amount of data per unit time that TCP transports during equilibrium (steady state), i.e. after the initial slow start phase. We refer to this state as TCP Equilibrium, and that the equalibrium throughput is the maximum achievable for the TCP connection(s).

There are many variables to consider when conducting a TCP throughput test and this methodology focuses on some of the most common parameters that should be considered such as:

- Path MTU and Maximum Segment Size (MSS)
- RTT and Bottleneck BW
- Ideal TCP Window (Bandwidth Delay Product)

- Single Connection and Multiple Connection testing

One other important note, it is highly recommended that traditional Layer 2/3 type tests are conducted to verify the integrity of the network before conducting TCP tests. Examples include RFC2544, iperf (UDP mode), or manual packet layer test techniques where packet throughput, loss, and delay measurements are conducted.

2. Goals of this Methodology

Before defining the goals of this methodology, it is important to clearly define the areas that are not intended to be measured or analyzed by such a methodology.

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

- The 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

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

In contrast to the above exclusions, the goals of this methodology are to define a method to conduct a structured, end-to-end assessment of sustained TCP performance within a managed business class IP network. A key goal is to establish a set of "best practices" that an engineer should apply when validating the ability of a managed network to carry end-user TCP applications.

Some specific goals are to:

- Provide a practical test approach that specifies the more well understood (and end-user configurable) TCP parameters such as Window size, MSS (Maximum Segment Size), # connections, and how these affect the outcome of TCP performance over a network.

- Provide specific test conditions (link speed, RTT, window size, etc.) and maximum achievable TCP throughput under TCP Equilbrium conditions. For guideline purposes, provide examples of these test conditions and the maximum achievable TCP throughput during the equilbrium state. Section 2.1 provides specific details concerning the definition of TCP Equilibrium within the context of this draft.

- Define two (2) basic metrics that can be used to compare the performance of TCP connections under various network conditions

- In test situations where the recommended procedure does not yield the maximum achievable TCP throughput result, this draft provides some possible areas within the end host or network that should be considered for investigation (although again, this draft is not intended to provide a detailed diagnosis of these issues)

2.1 TCP Equilibrium State Throughput

TCP connections have three (3) fundamental congestion window phases as documented in RFC2581. These states are:

- Slow Start, which occurs during the beginning of a TCP transmission or after a retransmission time out event

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

- Retransmission phase, which include Fast Retransmit (Tahoe) and Fast Recovery (Reno and New Reno). When a packet is lost, the Congestion avoidance phase transitions to a Fast Retransmission or Recovery Phase dependent upon the TCP implementation.

The following diagram depicts these states.

```
            |        ssthresh
   TCP      |           |
   Through- |           |       Equilibrium
   put      |           |\      /\/\/\/\/\  Retransmit          /\/\ ...
            |           | \    /         |  Time-out           /
            |           |  \  /          |  _______          _/
            |  Slow   _/    |/           | /       | Slow  _/
            | Start _/      Congestion   |/        |Start_/   Congestion
            |     _/         Avoidance   Loss      |   _/     Avoidance
            |   _/                       Event     | _/
            | _/                                   |/
            |/___________________________________________________________
                                                            Time
```

This TCP methodology provides guidelines to measure the equilibrium throughput which refers to the maximum sustained rate obtained by congestion avoidance before packet loss conditions occur (which would cause the state change from congestion avoidance to a retransmission phase). All maximum achievable throughputs specified in Section 3 are with respect to this Equilibrium state.

2.2 Metrics for TCP Throughput Tests

This draft focuses on a TCP throughtput methodology and also provides two 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 account for the myriad of variables (i.e. RTT variation, loss conditions, TCP implementation, etc.). However, these two basic metrics faciliate TCP throughput comparisons under varying network conditions and between network traffic management techniques.

The TCP Efficiency metric is the percentage of bytes that were not retransmitted and is defined as:

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

This metric provides a comparative measure between various QoS mechanisms such as traffic management, congestion avoidance, and also various TCP implementations (i.e. Reno, Vegas, etc.).

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

$$\frac{100{,}000 - 2{,}000}{100{,}000} \times 100 = 98\%$$

Note that the retranmitted bytes may have occurred more than once, and these multiple retransmissions are added to the bytes retransmitted count.

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

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

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

An example would be the bulk transfer of 100 MB upon 5 simultaneous TCP connections over a 500 Mbit/s Ethernet service (each connection uploading 100 MB). 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 be 12/8 = 1.5, which indicates that the transfer across all connections took 1.5 times longer than the ideal.

Note that both the TCP Efficiency and TCP Transfer Time metrics must be measured during each throughput test. The correlation of TCP Transfer Time with TCP Efficiency can help to diagnose whether the TCP Transfer Time was negatively impacted by retransmissions (poor TCP Efficiency).

3. TCP Throughput Testing Methodology

As stated in Section 1, it is considered best practice to verify the integrity of the network by conducting Layer2/3 stress tests such as RFC2544 (or other methods of network stress tests). If the network is not performing properly in terms of packet loss, jitter, etc. then the TCP layer testing will not be meaningful since the equalibrium throughput would be very difficult to achieve (in a "dysfunctional" network).

The following represents the sequential order of steps to conduct the TCP throughput testing methodology:

1. Identify the Path MTU. Packetization Layer Path MTU Discovery or PLPMTUD (RFC4821) should be conducted to verify the minimum network path MTU. Conducting PLPMTUD establishes the upper limit for the MSS to be used in subsequent steps.

2. Baseline Round-trip Delay and Bandwidth. These measurements provide estimates of the ideal TCP window size, which will be used in subsequent test steps.

3. TCP Connection Throughput Tests. With baseline measurements of round trip delay and bandwidth, a series of single and multiple TCP connection throughput tests can be conducted to baseline the network performance expectations.

4. Traffic Management Tests. Various traffic management and queuing techniques are tested in this step, using multiple TCP connections. Multiple connection testing can verify that the network is configured properly for traffic shaping versus policing, various queuing implementations, and 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 dedicated communications test instrument and these TCP test hosts be capable of emulating both a client and a server.

Whether the TCP test host is a standard computer or dedicated test instrument, the following areas should be considered when selecting a test host:

- TCP implementation used by the test host OS, i.e. Linux OS kernel using TCP Reno, TCP options supported, etc. This will obviously be more important when using custom test equipment where the TCP implementation may be customized or tuned to run in higher performance hardware

- Most importantly, the TCP test host must be capable of generating and receiving stateful TCP test traffic at the full link speed of the network under test. As a general rule of thumb, testing TCP throughput at rates greater than 100 Mbit/sec generally requires high performance server hardware or dedicated hardware based test tools.

- To measure RTT and TCP Efficiency per connection, this will generally require dedicated hardware based test tools. In the absence of dedciated hardware based test tools, these measurements may need to be conducted with packet capture tools (conduct TCP throughput tests and analyze RTT and retransmission results with packet captures).

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 link, 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 in real world situations.

Packetization Layer Path MTU Discovery or PLPMTUD (RFC4821) should be conducted to verify the minimum 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 the TCP protocol. RFC4821 specifies a search_high and search_low parameter for the MTU. As specified in RFC4821, a value of 1024 is a generally safe value to choose for search_low in modern networks.

It is important to determine the overhead of the links in the 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 nominal window size (8KB, etc.). The window size should be kept small to minimize the possibility of congesting the network, which could induce congestive 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 byte 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 process and reduce the advertised MSS to avoid fragmentation. This is certainly a desirable feature from a network perspective, but 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 byte MTU. In this example, the test client would 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 would be to reduce the probe size to half between the unsuccessful search_high and successful search_low value, and increase by increments of 1/2 when seeking the upper limit.

3.2. Baseline Round-trip Delay and Bandwidth

Before stateful TCP testing can begin, it is important to baseline the round trip delay and bandwidth of the network to be tested. These measurements provide estimates of the ideal TCP window size, which will be used in subsequent test steps. These latency and bandwidth tests should be run during the time of day for which the TCP throughput tests will occur.

The baseline RTT is used to predict the bandwidth delay product and the TCP Transfer Time for the subsequent throughput tests. Since this methodology requires that RTT be measured during the entire throughput test, the extent by which the RTT varied during the throughput test can be quantified.

3.2.1 Techniques to Measure Round Trip Time

Following the definitions used in the references of the appendix; Round Trip Time (RTT) is the time elapsed between the clocking in of the first bit of a payload packet to the receipt of the last bit of the corresponding acknowledgement. Round Trip Delay (RTD) is used synonymously to twice the Link Latency.

In any method used to baseline round trip delay between network end-points, it is important to realize that network latency is the sum of inherent network delay and congestion. The RTT should be baselined during "off-peak" hours to obtain a reliable figure for network latency (versus additional delay caused by congestion).

During the actual sustained TCP throughput tests, it is critical to measure RTT along with measured TCP throughput. Congestive effects can be isolated if RTT is concurrently measured.

This is not meant to provide an exhaustive list, but summarizes some of the more common ways to determine round trip time (RTT) through the network. The desired resolution of the measurement (i.e. msec versus usec) may dictate whether the RTT measurement can be achieved with standard tools such as ICMP ping techniques or whether specialized test equipment would be required with high 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 end-end. This test equipment RTT measurement may be compatible with delay measurement protocols specified in RFC5357.

- Conduct packet captures of TCP test applications using for example "iperf" or FTP, etc. By running multiple experiments, the packet captures can be studied to estimate RTT based upon the SYN -> SYN-ACK handshakes within the TCP connection set-up.

- ICMP Pings may also be adequate to provide round trip time estimations. Some limitations of ICMP Ping are the msec resolution and whether the network elements respond to pings (or block them).

3.2.2 Techniques to Measure End-end Bandwidth

There are many well established techniques available to provide estimated measures of bandwidth over a network. This measurement should be conducted in both directions of the network, especially for access networks which are inherently asymmetrical. Some of the asymmetric implications to TCP performance are documented in RFC-3449 and the results of this work will be further studied to determine relevance to this draft.

The bandwidth measurement test must be run with stateless IP streams (not stateful TCP) in order to determine the available bandwidth in each direction. And this test should obviously be performed at various intervals throughout a business day (or even across a week). Ideally, the bandwidth test should produce a log output of the bandwidth achieved across the test interval AND the round trip delay.

And during the actual TCP level performance measurements (Sections 3.3 - 3.5), the test tool must be able to track round trip time of the TCP connection(s) during the test. Measuring round trip time variation (aka "jitter") provides insight into effects of congestive delay on the sustained throughput achieved for the TCP layer test.

3.3. TCP Throughput Tests

This draft specifically defines TCP throughput techniques to verify sustained TCP performance in a managed business network. Defined in section 2.1, the equalibrium throughput reflects the maximum rate achieved by a TCP connection within the congestion avoidance phase on a end-end network path. This section and others will define the method to conduct these sustained throughput tests and guidelines of 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 can be conducted to baseline network performance against expectations.

It is recommended to run the tests in each direction independently first, then run both directions simultaneously. In each case, the TCP Efficiency and TCP Transfer Time metrics must be measured in each direction.

3.3.1 Calculate Optimum TCP Window Size

The optimum TCP window size can be calculated from the bandwidth delay product (BDP), which is:

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

By dividing the BDP by 8, the "ideal" TCP window size is calculated. An example would be a T3 link with 25 msec RTT. The BDP would equal ~1,105,000 bits and the ideal TCP window would equal ~138,000 bytes.

The following table provides some representative network link speeds, latency, BDP, and associated "optimum" TCP window size. Sustained TCP transfers should reach nearly 100% throughput, minus the overhead of Layers 1-3 and the divisor of the MSS into the window.

For this single connection baseline test, the MSS size will effect the achieved throughput (especially for smaller TCP window sizes). Table 3.2 provides the achievable, equalibrium TCP throughput (at Layer 4) using 1460 byte MSS. Also in this table, the case of 58 byte L1-L4 overhead including the Ethernet CRC32 is used for simplicity.

Table 3.2: Link Speed, RTT and calculated BDP, TCP Throughput

Link			Ideal TCP	Maximum Achievable
Speed*	RTT (ms)	BDP (bits)	Window (kbytes)	TCP Throughput(Mbps)
T1	20	30,720	3.84	1.17
T1	50	76,800	9.60	1.40
T1	100	153,600	19.20	1.40
T3	10	442,100	55.26	42.05
T3	15	663,150	82.89	42.05
T3	25	1,105,250	138.16	41.52
T3(ATM)	10	407,040	50.88	36.50
T3(ATM)	15	610,560	76.32	36.23
T3(ATM)	25	1,017,600	127.20	36.27
100M	1	100,000	12.50	91.98
100M	2	200,000	25.00	93.44
100M	5	500,000	62.50	93.44
1Gig	0.1	100,000	12.50	919.82
1Gig	0.5	500,000	62.50	934.47
1Gig	1	1,000,000	125.00	934.47
10Gig	0.05	500,000	62.50	9,344.67
10Gig	0.3	3,000,000	375.00	9,344.67

* Note that link speed is the minimum link speed throughput a network; i.e. WAN with T1 link, etc.

Also, the following link speeds (available payload bandwidth) were used for the WAN entries:

- T1 = 1.536 Mbits/sec (B8ZS line encoding facility)
- T3 = 44.21 Mbits/sec (C-Bit Framing)
- T3(ATM) = 36.86 Mbits/sec (C-Bit Framing & PLCP, 96000 Cells per second)

The calculation method used in this document is a 3 step process :

1 - We determine what should be the optimal TCP Window size value based on the optimal quantity of "in-flight" octets discovered by the BDP calculation. We take into consideration that the TCP Window size has to be an exact multiple value of the MSS.

2 - Then we calculate the achievable layer 2 throughput by multiplying the value determined in step 1 with the MSS & (MSS + L2 + L3 + L4

    Overheads) divided by the RTT.
3 - Finally, we multiply the calculated value of step 2 by the MSS
    versus (MSS + L2 + L3 + L4 Overheads) ratio.

This gives us the achievable TCP Throughput value. Sometimes, the maximum achievable throughput is limited by the maximum achievable quantity of Ethernet Frames per second on the physical media. Then this value is used in step 2 instead of the calculated one.

The following diagram compares achievable TCP throughputs on a T3 link with Windows 2000/XP TCP window sizes of 16KB versus 64KB.

```
           45|
             |          _____42.1M
           40|          |64K|
TCP          |          |   |
Throughput 35|          |   |           _____34.3M
in Mbps      |          |   |           |64K|
           30|          |   |           |   |
             |          |   |           |   |
           25|          |   |           |   |
             |          |   |           |   |
           20|          |   |           |   |           _____20.5M
             |          |   |           |   |           |64K|
           15| 14.5M____|   |           |   |           |   |
             |      |16K|   |           |   |           |   |
           10|      |   |   |   9.6M+---+   |           |   |
             |      |   |   |       |16K|   |   5.8M____+   |
            5|      |   |   |       |   |   |       |16K|   |
             |______+___+___+_______+___+___+_______+__ +___+_______
                        10              15              25
                                RTT in milliseconds
```

The following diagram shows the achievable TCP throughput on a 25ms T3 when the TCP Window size is increased and with the RFC1323 TCP Window scaling option.

```
           45|
             |                                              +-----+42.47M
           40|                                              |     |
TCP          |                                              |     |
Throughput 35|                                              |     |
in Mbps      |                                              |     |
           30|                                              |     |
             |                                              |     |
           25|                                              |     |
             |                                ______ 21.23M |     |
           20|                                |    |        |     |
             |                                |    |        |     |
           15|                                |    |        |     |
             |                                |    |        |     |
           10|               +----+10.62M     |    |        |     |
             |  _______5.31M |    |           |    |        |     |
            5|  |     |      |    |           |    |        |     |
             |__+_____+______+____+___________+____+________+_____+___
                   16           32           64            128
                               TCP Window size in KBytes
```

3.3.2 Conducting the TCP Throughput Tests

There are several TCP tools that are commonly used in the network world and one of the most common is the "iperf" tool. With this tool, hosts are installed at each end of the network segment; one as client and the other as server. The TCP Window size of both the client and the server can be maunally set and the achieved throughput is 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 also become part of the window size / throughput characterization.

Host hardware performance must be well understood before conducting the TCP throughput tests and other tests in the following sections. Dedicated test equipment will generally be required, especially for line rates of GigE and 10 GigE.

The TCP throughput test should be run over a a long enough duration to properly exercise network buffers and also characterize performance during different time periods of the day. The results must be logged at the desired interval and the test must record RTT and TCP retransmissions at each interval.

This correlation of retransmissions and RTT over the course of the test will clearly identify which portions of the transfer reached

TCP Equilbrium state and to what effect increased RTT (congestive effects) may have been the cause of reduced equilibrium performance.

Additionally, the TCP Efficiency and TCP Transfer time metrics should be logged in order to further characterize the window size tests.

3.3.3 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 window sizes configured in the end-user environment. For example, if the BDP for a long-fat pipe turns out to be 2MB, then it is probably more realistic to test this pipe with multiple connections. Assuming typical host computer window settings of 64 KB, using 32 connections would realistically test this pipe.

The following table is provided to illustrate the relationship of the BDP, window size, and the number of connections required to utilize the the available capacity. For this example, the network bandwidth is 500 Mbps, RTT is equal to 5 ms, and the BDP equates to 312 KBytes.

Window	#Connections to Fill Link
16KB	20
32KB	10
64KB	5
128KB	3

The TCP Transfer Time metric is useful for conducting multiple connection tests. Each connection should be configured to transfer a certain payload (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 example table listed above, the 64KB window is considered. Each of the 5 connections would be configured to transfer 100MB, and each TCP should obtain a maximum of 100 Mb/sec per connection. 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 for these conditions).

Additionally, the TCP Efficiency metric should be computed for each connection tested (defined in section 2.2).

3.3.4 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 setting. For cases where the sustained TCP throughput does not equal the predicted value, some possible causes are listed:

- Network congestion causing packet loss; the TCP Efficiency metric is a useful gauge to compare network performance
- Network congestion not causing packet loss but increasing RTT
- Intermediate network devices which actively regenerate the TCP connection and can alter window size, MSS, etc.
- Over utilization of available link or rate limiting (policing). More discussion of 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 of the 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, etc.)

Configuring the end-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 for 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 test equipment will generally be required, especially for line rates of GigE and 10 GigE.

3.4.1 Traffic Shaping Tests

For services where the available bandwidth is rate limited, there are two (2) techniques used to implement rate limiting: traffic policing and 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 exceeded).

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 given available bandwidth. Through this section, the available 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 connection test. Proper shaping will provide a fair distribution of the available bottleneck bandwidth, while traffic policing will not.

The traffic shaping tests build upon the concepts of multiple connection testing as defined in section 3.3.3. Calculating the BDP for the bottleneck bandwidth is first required and then selecting the number of connections / window size 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 window size evenly fill the bottleneck bandwidth (about 100 Mbps per connection).

The traffic shaping should be run over a long enough duration to properly exercise network buffers 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 Efficiency and TCP Transfer time metric. Additionally, it is recommended to log RTT and retransmissions per connection over the test interval.

3.4.1.1 Interpretation of Traffic Shaping Test Restults

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 would not be fair and the resulting throughput plot would reveal "spikey" connection throughput consumption of the competing TCP connections (due to the 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 can benefit greatly from RED based techniques. Without RED, TCP is generally not able to achieve the full bandwidth of the bottleneck interface. With RED enabled, TCP congestion avoidance throttles the connections on the higher speed interface (i.e. LAN) and can reach equalibrium with the bottleneck bandwidth (achieving closer to full throughput).

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

The RED tests also build upon the concepts of multiple connection testing as defined in secion 3.3.3. Calculating the BDP for the bottleneck bandwidth is first required and then selecting the number of connections / 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. In this example, 10 connections would produce TCP bursts of 64KB for each connection. If the timing of the TCP tester permits, these TCP bursts could stress queue sizes in the 512KB range. Again experimentation will be required and the proper number of TCP connections / window size will be dictated by the size 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 will 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 can 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 would be greatly reduced (generally 20-40%) and the TCP Transfer time would be proportionally longer then the ideal transfer time.

Additionally, the TCP Transfer Efficiency metric is useful, since non-RED implementations will exhibit a lower TCP Tranfer Efficiency than RED implementations.

4. Acknowledgements

The author would like to thank Gilles Forget, Loki Jorgenson, and Reinhard Schrage for technical review and original contributions to this draft-03.

Also thanks to Matt Mathis and Matt Zekauskas for many good comments through email exchange and for pointing us to great sources of information pertaining to past works in the TCP capacity area.

5. References

[RFC2581] Allman, M., Paxson, V., Stevens W., "TCP Congestion Control", RFC 2581, June 1999.

[RFC3148] Mathis M., Allman, M., "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.

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

[RFC3449] Balakrishnan, H., Padmanabhan, V. N., Fairhurst, G., Sooriyabandara, M., "TCP Performance Implications of Network Path Asymmetry", RFC 3449, December 2002

[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

[MSMO] The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm Mathis, M.,Semke, J, Mahdavi, J, Ott, T July 1997 SIGCOMM Computer Communication Review, Volume 27 Issue 3

[Stevens Vol1] TCP/IP Illustrated, Vol1, The Protocols Addison-Wesley

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

Loki Jorgenson
nooCore

Phone: (604) 908-5833
ljorgenson@nooCore.com

Reinhard Schrage
Schrage Consulting

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