# Netest: A Tool to Measure the Maximum Burst Size, Available Bandwidth and Achievable Throughput

Guojun Jin Brian Tierney

*Distributed Systems Department*
*Lawrence Berkeley National Laboratory*
*1 Cyclotron Road, Berkeley, CA 94720*
*g_jin@lbl.gov*

## Abstract

Distinguishing available bandwidth and achievable throughput is essential for improving network application performance. Achievable throughput is the throughput considering a number of factors such as network protocol, host speed, network path, and TCP buffer space, whereas available bandwidth only considers the network path. Without understanding this difference, trying to improve network application performance is like the "blind men feeling the elephant" problem [4]. In this paper, we define and differentiate *bandwidth* and *throughput*, and discuss which part of each is *achievable* and which is *available*. Also, we introduce and discuss a new concept, Maximum Burst Size, that is crucial to obtaining good network performance. A new tool, *netest*, is introduced which is designed to help users to determine the *available* bandwidth. It provides information to *achieve* better throughput while fairly sharing the available bandwidth, thus reducing misuse of the network.

## I. Introduction

Available bandwidth is obtained via a simple arithmetic operation: capacity minus utilization. However, this is not clear to many developers nor users of networked applications. After performing TCP tuning techniques, such as those explained in [8], a user or developer might think that they have fully utilized available bandwidth. However it is quite possible that they are only utilizing the bandwidth that their application is capable of achieving, and not all the available bandwidth. For example, if a store has three bottles of beer, after one bottle of beer is sold, the available beer is two bottles. However, one might argue that the available beer is three bottles because someone could possibly grab all three bottles of beer before the buyer leaves the store. This demonstrates achievement v.s. availability. It is important that what is *available* and what is *achievable* must be differentiated and defined clearly.

When the available bandwidth is unknown, the best network performance is obtained via adaptive control mechanisms such as TCP. However, TCP often does not perform well in high bandwidth long delay paths, due to the fact that it recovers very slowly from packet loss [9]. Techniques to get better performance over high-speed networks include reliable UDP-based methods [3][2][10], and using parallel TCP streams [1][11]. Without knowing the Maximum Burst Size (MBS — the maximum number of bytes that can travel through a network path without causing packet loss), both of those mechanisms potentially violate the fairly sharing policy of network resources. This paper shows how knowledge of the MBS can aid in optimal and fair use of the network.

In this paper, we explain what is available and what is achievable in the network, we then discuss some real use cases where applications were achieving much lower than expected performance based on the available network bandwidth. We introduce *netest* (pronounced "net"-"est", short for network estimator), which is designed to provide information about each element on a path between two end hosts. This information includes the available bandwidth of the path and/or the maximum achievable throughput between the two end hosts. *Netest* can help to identify the source of poor network performance such as a problematic router, sending host, receiving host, lack of TCP buffers, etc. *Netest* also provides advice on what one can do to improve application throughput. We use *netest* in a case study and show how to interpret *netest* output.

## II. Terminology

In this section, we distinguish available and achievable, bandwidth and throughput, and provide the definitions of available bandwidth, achievable throughput and maximum throughput.

> **Bandwidth** — the speed that a network element can forward traffic. It has two characteristics — physical and available, and both of them are independent of end hosts and protocol type.

• *Physical bandwidth*, or *capacity* (*C*), is the maximum number of bits per second a network element can transfer. The physical bandwidth of an end-to-end path is determined by the slowest network element along the path.

• *Utilization* (*U*) is the percentage of capacity currently being consumed by aggregated traffic on a link or path:

$$U = \frac{Traffic}{C}$$

• *Available bandwidth* (*A*) is the capacity minus utilization over a given time interval. This is applicable to paths, links, routers or switches.

$A(t_s, t_e)$ = Capacity - Traffic

$= C \times (1 - U)$

$\neq A(T_{window})$

$T_{window} = t_s - t_e$

$t_s$ is the time the measurement starts

$t_e$ is the time the measurement ends

**Throughput** — amount of data that is successfully sent from one host to another via a network. It may be limited by every component along the path from source host to destination host, including all hardware and software. Throughput also has two characteristics — achievable and maximum.

• *Maximum throughput* is the best transfer rate that can be successfully performed between two end hosts if they are connected back-to-back.

• *Achievable throughput* is the throughput between two end points under a completely given set of conditions, such as transmission protocol, end host hardware, operating system, tuning method and parameters, etc. This characteristic represents the performance that an application in this specific setting might achieve. Since the bottleneck could be in an end host, achievable throughput may or may not correlate with available bandwidth.

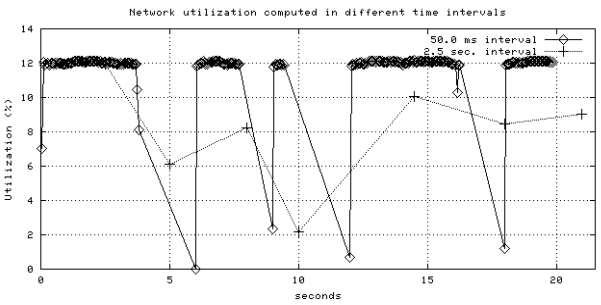


Figure 1 Available bandwidth at 2 different time intervals

Measurement of available bandwidth depends strongly on the time interval used for the measurement. For example, Figure 1 shows the network utilization measured with both 50 ms and 2.5s time intervals. Available bandwidth is not, however, an indication of what an application can actually expect to obtain. For this, achievable throughput is the proper measurement. To make this clear, we examine the behavior when UDP traffic competes with TCP. If a path has N% of its capacity used by TCP traffic, the available bandwidth is (100-N)%. When a UDP stream comes in at a rate of 90% line speed, sooner or later, the UDP traffic will gain the 90% of total bandwidth, thus causing the TCP throughput to drop to 10% of the capacity. That is, UDP achieved the maximum throughput by aggressively taking the other application's bandwidth, while TCP achieved lower throughput because of its congestion control algorithm. The question is how to make a protocol like TCP utilize the available bandwidth without unfairly disturbing current traffic and without losing desired bandwidth. The answer is to use the maximum burst size (MBS) as illustrated by the following case study.

The *maximum burst size* is the maximum number of bytes a router can absorb without dropping a packet. This is determined by the size of the router queue, and by the current cross traffic at that router. Not exceeding the MBS is the key to obtaining good achievable throughput.

## III. Case Study

In this section, we analyze a couple of typical cases where TCP and even UDP applications failed to obtain desired performance on the high-speed network.

### *A. Maximum Burst Size (MBS)*

When tuning TCP connections, a common technique is to set the TCP buffer size to the size of the bandwidth-delay product (BDP) of the path. For example, for a 1Gb/s path with 80 ms RTT, one might set the TCP buffers to 10 MB, which allows the host to send a burst of packets up to that size. But, if the MBS of the bottleneck router is only 2MB, then packets would get dropped if the window was all the way open.

This typical example was an 8-hop network path with 30 ms delay, where the bottleneck link was OC-12 (622 Mb/s). This was not a long delay path, but TCP throughput was less than 10 Mb/s, and UDP throughput was around 80 Mb/s. The utilization reported by SNMP was between 5-10% at the bottleneck link. We ran *netest* on this path with 660 Mb/s sending rate and found the maximum burst size was only 82K bytes. *Netest* suggested to run pipechar [6] with option "-Q" to determine where is the bottleneck. Pipechar showed that router 7 (OC-12) caused this short queue behavior. In this circumstance, use of the bandwidth

delay product will set TCP congestion window to

$$0.03s \times 622 \times 10^6 b/s \;=\; 18.66Mbits \;=\; 2.3325MBytes$$

This allows TCP to send a large burst which exceeds the MBS, thus causing the narrow-link router to drop packets when cross traffic exists, which was almost always true. The TCP congestion window then will shutdown and the maximum throughput will be limited to

$$\frac{82KB \times 8bits/Byte}{0.03s} \;=\; 21.8667Mb/s$$

Due to retransmissions the actual achievable throughput was even lower.

UDP throughput also depends on the burst size and rate of the traffic. An 82KB effective queue of the bottleneck router implies that the maximum burst duration (at line speed, 622 Mb/s in this case) for both probe traffic and cross traffic to avoid overflow is:

$$\frac{MBS}{LineSpeed} \;=\; \frac{82KB}{622Mb/s} \;=\; 1.05466ms$$

because when a burst leaves the previous router, it will travel at link (line) speed. Since we can characterize all cross traffic as one aggregated stream, each stream (aggregated cross traffic and measurement traffic) has a minimum 1.05466-ms safety burst interval (gap for router to drain two 82KB bursts on average according to the effective queue size) to not overflow the smallest queue on this path. Without knowing how to control the burst, UDP throughput will vary depending on how many cross traffic bursts have a duration longer than 1.05466 ms (exceed 82KB in length). The more big bursts of cross traffic, the lower the UDP throughput will be.

Burst control can help to increase throughput under these circumstances. Two burst sizes were chosen to illustrate this issue. One UDP stream had a 160KB burst size (twice as big as the safe burst length to cause average 50% packet drop) and 20 ms burst period (context switch takes slightly over 10 ms, so rounded up to 20 ms for easy computation). The other UDP stream had 40KB burst (half the size of the safety burst length to minimize the packet drop, its burst duration — 501.5 μs at this line speed — is approximately one half of the burst period to allow the router to drain the traffic), and 1 ms burst period to get the maximum throughput. In the first stream, 50 bursts can be sent every second, so the maximum transfer rate should be

$$160KB \times 8bits/Byte \times 50/s \;=\; 64Mb/s$$

and 32 Mb/s was expected because of the predicted 50% packet drop rate. In the second stream, the maximum throughput should be 320Mb/s since 1000 bursts can be transferred in one second. In 30 tests for each stream, the 160KB burst stream gave a throughput range of 44~49 Mb/s (it is assumed that not all bursts had encountered a large cross traffic burst, so the throughput is higher). The 40KB burst stream achieved 300 Mb/s throughput on this path, which is 7% below the expectation. This is reasonable due to context switch time and possible packet drops. For confirmation, network engineers proposed an experimental solution: they added a switch with an 8M-byte queue in front of router 7 as a buffer that temporarily solved the problem.

This case and further study [11] show that if burst size can be controlled properly, a simple reliable protocol may be built to replace UDP and TCP to achieve optimum throughput, utilize, and fairly share the available bandwidth with current traffic.

### *B. Parallel stream TCP*

Parallel stream TCP is used to overcome the problem that a single TCP stream recovers slowly from loss on high bandwidth delay product networks. This happens because TCP congestion window is not large enough to fill up the entire pipe. Because TCP streams have independent congestion windows, using parallel stream TCP can make the aggregated congestion window large, thus, producing higher aggregated TCP throughput. However, the parallel TCP streams may grab bandwidth from other TCP traffic, allowing unfair use of bandwidth [5].

Figure 2 shows two cases using parallel stream TCP on the same path — the top one is between two fast hosts, and the lower one is from an improperly patched Solaris host. The top line in the graph shows that if the number of parallel streams is more than necessary, it will cause congestion on the network and reduce the performance. It shows that two parallel TCP streams had utilized the available bandwidth because the aggregated throughput was less than double the single stream TCP throughput. 3 to 4 streams had more competition with other traffic, but much less gain in throughput. 5 to 8 streams aggressively took bandwidth from other traffic to maximize the throughput, and even more streams not only competed with other traffic, but also competed with each other, thus causing performance degradation.

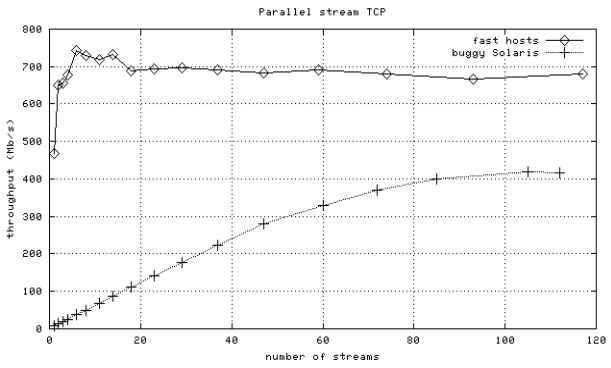


Figure 2 parallel stream TCP throughput

```
--- General statistics ---
Average bandwidth used in testing: 5.5436 Mb/s in 171.6751 sec
TCP transf rate:       min 5.6854 / avg 5.9354 / max 43.2980 Mb/s

--- General section ---
      Round Trip Time (RTT):        78.8000 ms
      Maximum burst size:           6709248 Bytes

--- UDP section ---
      Single stream UDP throughput: 347.3039 Mbps
      Multi- stream UDP throughput: 347.3039 Mbps
      Use of multiple streams is not recommended

--- TCP section ---
    Optimal TCP Window size:      766464 Bytes
      Single stream TCP throughput:  6.1854 Mbps
         Use of parallel TCP streams:      is recommended
        65 TCP streams can fully utilize the Available BW (estimated)

--- Advisory section ---
      Kernel may need patch
      Please rerun test with -FullD
maximum throughput is around 391.624 Mb/s limited by local kernel
```

Figure 3 automatic mode of netest from SLAC to ORNL

The lower line in the graph (Solaris) presented an amazingly smooth curve up to 112 parallel TCP streams to obtain the maximum throughput. This is highly unusual. Tuning techniques did not help to increase any single TCP throughput. The default behavior was measured by netest in Figure 3 (notice that advised number of TCP streams is 65, not 112. The reason is explained in Figure 5). The advisory section extrapolated what could be the cause of such behavior.

Figure 4, shows the results of running netest using the suggested "-FullD" option, which sets the receive buffer to be as large as the send buffer on the sending host. (This option was added to detect a problem in Solaris that we have observed on and off over the past several years. In some versions of Solaris, it appears that the TCP send buffers are not increased unless the TCP receive buffer size on the sending host is also increased, even when no traffic is traveling in that direction). This increased achievable throughput by a factor of 22. This problem was only observed on one host on the tested subnet, and was not a problem from a different Solaris host to the same destination. It is likely that this is an issue with this version of Solaris on this host. Possible reasons to explain why the single stream TCP performance is still low are described in a paper by Hacker [5].

```
...
--- TCP section ---
      Optimal TCP Window size:      2254493 Bytes
      Single stream TCP throughput:  136.7045 Mbps
         Use of parallel TCP streams:      is recommended
           3 TCP streams can fully utilize the Available bandwidth
```

Figure 4 Use same bigger buffer for both Tx/Rx on the same host

These examples depict that the parallel stream technique must be used properly with knowledge of how parallel streams can improve performance. Improper use of parallel streams can cause unfair use of the network, even though it maximizes the achievable throughput. Too many streams also waste resources, for example using 112

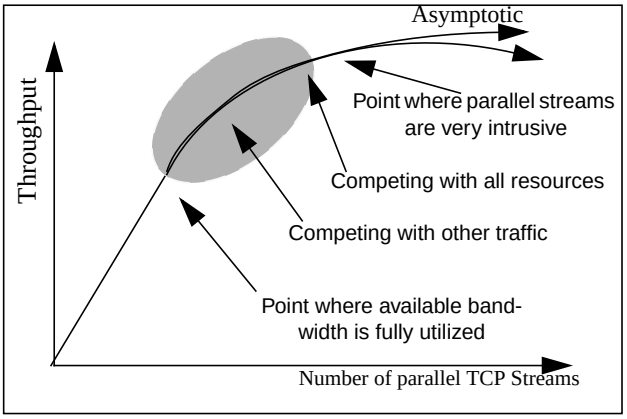


Figure 5 Parallel TCP throughput

streams to achieve the maximum achievable throughput where 3 streams is adequate. The proper use of parallel TCP streams is illustrated in Figure 5. The optimal number of parallel streams is at the lower intersection of the edge of the shaded area and the curve. The number of streams within the shaded area may increase the achievable throughput, but begin to disturb other traffic.

To summarize, achievable throughput is the most useful characteristic for applications to base their end-to-end performance expectations. Using it in place of available bandwidth helps avoid unnecessary effort spent attempting to tune systems and applications beyond what is possible.

## IV. NETEST

*Netest* is designed to measure achievable throughput and available bandwidth in a minimally intrusive manner. It can measure available bandwidth accurately in a few seconds [7]. Netest provides useful information which TCP applications can use to improve their performance and achievable throughput without requiring network expertise. The basic usage is very simple:

remote_server% *netest*

local_client% *netest* -t remote_server [-abw]

Sample *netest* results are shown in Figure 6, which demonstrates a common case. This path has a lot of available bandwidth, but an application using single stream TCP with tuned TCP buffers of 6 MB only achieved 475.5 Mb/s achievable throughput. This is because to achieve 800 Mb/s TCP throughput requires a 16 MB congestion window. Therefore, using 2 parallel TCP streams will fully utilize the available bandwidth on this path.

*Netest* output has several sections:

- The verbose section (shown in Figure 6) shows the hardware information of both end hosts, average time spent involving major system functions on the local host.

```
local CPU speed 1394.10 MHz AMD
usable local mem   250 MBytes
mem bus speed      512 MBps
max mem copy speed  320 MBps
min mem copy speed  295 MBps
NIC speed      814.288 Mb/s
sub_cost         5 ns          — time for entering and exiting a subroutine call
 ts_call         283 ns        — Unix gettimeofday function
sys-call         213 ns        — Unix system call

Remote server is Ver2 RC-10 2003-01-02
Linux Host-97-16 2.4.19-net100
has 1 thread running at Func0
usable remote mem   883 MBytes
mem bus speed      785 MBps
max mem copy speed  514 MBps
min mem copy speed  496 MBps
NIC speed      1000.500 Mb/s

--- General statistics ---
Average bandwidth used in testing: 7.7032 Mb/s in 112.5857 sec
TCP transf rate:    min 22.2157 / avg 180.5447 / max 804.0995 Mb/s

--- General section ---
      Round Trip Time (RTT):        158.9000 ms
      Maximum burst size:           6029312 Bytes

--- UDP section ---
      Single stream UDP throughput:   719.7197 Mbps
      Multi- stream UDP throughput:   796.5154 Mbps
        Use of multiple streams is not recommended

--- TCP section ---
      Optimal TCP Window size:      6029312 Bytes
      Single stream TCP throughput:   475.5225 Mbps
         Use of parallel TCP streams:       is recommended
            2 TCP streams can fully utilize the Available bandwidth

Max throughput is around 804.1 Mb/s limited by local memory
```

Figure 6 Netest result on a long delay path cross country

- The statistics section tells how long *netest* ran to get the result, and how much bandwidth was used. Also, it tells the minimum, average, and the maximum TCP throughputs that were achieved during this test.
- The general section reports the round trip time (RTT) and the maximum burst size (MBS). The MBS is the maximum number of bytes that can be sent continuously from the source host to the destination host without causing any of network elements along the path, including the receiving host, to drop a packet. This is an important characteristic missing from most other tools.
- The UDP section reports the *achievable throughput* of single stream UDP. Based throughput for a single UDP stream and the hardware information, *netest* may also measure 2-parallel UDP streams to determine if the bottleneck is at the kernel of the sending/receiving host.
- The TCP section contains the information for tuning TCP-based application performance. It provides information on two primary TCP categories: optimal TCP window size (sending/receiving buffer space), and the expected single stream TCP throughput. *Netest* can also report the number of parallel TCP streams required to fully utilize the available bandwidth, and the number of parallel TCP streams where it starts to become intrusive to other traffic on the network.
- The Advisory section provides additional information for solving problems that cause poor performance. For example, if a router does not allow IP fragmentation, or is misconfigured in some way.

## V. Conclusion

This paper shows that distinguishing available bandwidth from achievable throughput is crucial to fully utilize the available bandwidth without interfering with other traffic. We showed how the MBS is very important to avoid dropping packets, and achieve good performance. The *netest* tool can be used to help tune network applications, analyze network problems, and provide useful information on how to resolve problems. It is easy to use and less intrusive than many similar tools. It is a useful tool to help users understand fair use of available bandwidth in order to maximize achievable throughput.

## VI. Acknowledgments

This work was supported by the Director, Office of Science. Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division under U.S. Department of Energy Contract No. DE-AC03-76SF00098. This is report no. LBNL-48350

## VII. References

[1] H. Sivakumar, Stuart Bailey, and Robert L. Grossman. “PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks”. In IEEE Supercomputing 2000.

[2] H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, Q. Zhang, “Simple Available Bandwidth Utilization Library for High speed Wide Area Networks”, Resource: http://www.dataspaceweb.net/sabul.htm

[3] Tom Dunigan, “Almost TCP over UDP (ATOU)”, Available: http://www.csm.ornl.gov/~dunigan/netperf/atou.html

[4] Jill Gemmill, “Blind men felling the elephant managing application network performance: standards, tools and challenges”, Integrated Design and Process Technology Vol. 1, p. 63-69, June 2001

[5] T. J. Hacker, Brian D. Athey, The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network, Aug. 2001.

[6] G. Jin, G. Yang, B. Crowley, D. Agarwal, “Network Characterization Service (NCS)”, HPDC-10 Symposium, August 2001

[7] G. Jin, “Algorithms and Requirements for Measuring Network Bandwidth”, LBNL Report: LBNL-48330

[8] B. Tierney, “TCP tuning guide for distributed application on wide area networks”, USENIX & SAGE Login, vol. 26, No. 1, Feb. 2001

[9] Sally Floyd, “HighSpeed TCP for Large Congestion Windows”, Available: http://www.icir.org/floyd/hstcp.html

[10] S. Wallace, “Tsunami: A Hybrid TCP/UDP based file transfer protocol”, Available: http://www.ncne.nlanr.net/training/techs/2002/0728/presentations/200207-wallace1_files/v3_document.htm

[11] D. Agarwal, J. González, G. Jin, B. Tierney, “An Infrastructure for Passive Network Monitoring of Application Data Streams”, PAM, April 2003