“In the name of God”

# Iperf

Iperf –s: command for server

Iperf –c <server ip>: command for client

Calculate BW exp :9.4Gb/s

iPerf3 is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, buffers and protocols (TCP, UDP, SCTP with IPv4 and IPv6). For each test it reports the bandwidth, loss, and other parameters. This is a new implementation that shares no code with the original iPerf and also is not backwards compatible. iPerf was orginally developed by [NLANR/DAST](https://iperf.fr/contact.php#authors). iPerf3 is principally developed by [ESnet](https://www.es.net/" \t "_blank) / [Lawrence Berkeley National Laboratory](https://www.lbl.gov/). It is released under a three-clause [BSD license](https://en.wikipedia.org/wiki/BSD_licenses).

It is accurate but UDP,TCP

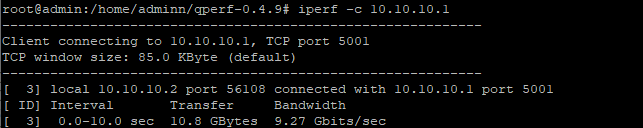
**iPerf features**

* TCP and [SCTP](https://en.wikipedia.org/wiki/Stream_Control_Transmission_Protocol)
  + Measure bandwidth
  + Report MSS/MTU size and observed read sizes.
  + Support for TCP window size via socket buffers.
* UDP
  + Client can create UDP streams of specified bandwidth.
  + Measure packet loss
  + Measure [delay jitter](https://en.wikipedia.org/wiki/Packet_delay_variation)
  + Multicast capable
* Cross-platform: Windows, Linux, Android, MacOS X, FreeBSD, OpenBSD, NetBSD, [VxWorks](https://en.wikipedia.org/wiki/VxWorks), Solaris,...
* Client and server can have multiple simultaneous connections (-P option).
* Server handles multiple connections, rather than quitting after a single test.
* Can run for specified time (-t option), rather than a set amount of data to transfer (-n or -k option).
* Print periodic, intermediate bandwidth, jitter, and loss reports at specified intervals (-i option).
* Run the server as a daemon (-D option)
* Use representative streams to test out how link layer compression affects your achievable bandwidth (-F option).
* A server accepts a single client simultaneously (iPerf3) multiple clients simultaneously (iPerf2)
* New: Ignore TCP slowstart (-O option).
* New: Set target bandwidth for UDP and (new) TCP (-b option).
* New: Set IPv6 flow label (-L option)
* New: Set congestion control algorithm (-C option)
* New: Use SCTP rather than TCP (--sctp option)
* New: Output in JSON format (-J option).
* New: Disk read test (server: iperf3 -s / client: iperf3 -c testhost -i1 -F filename)
* New: Disk write tests (server: iperf3 -s -F filename / client: iperf3 -c testhost -i1)

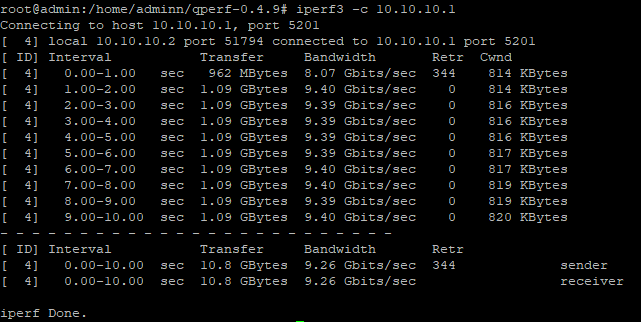
API for Iperf : python, C++

<https://github.com/esnet/iperf>

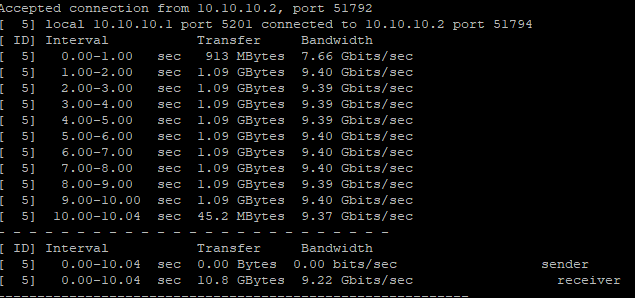
Client:



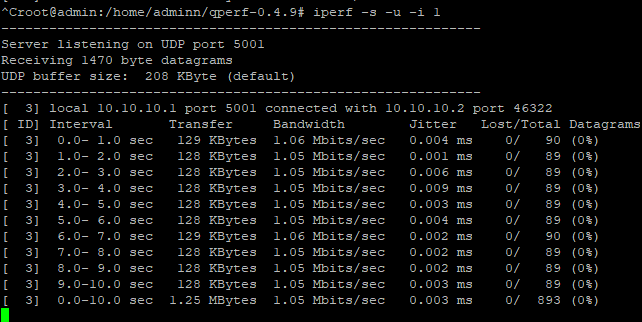
Client:



Server

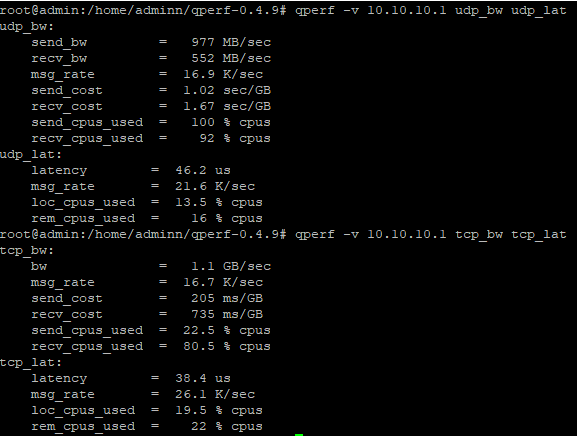


Server



# Qperf

TCP and UDP: Measure BW and Latency

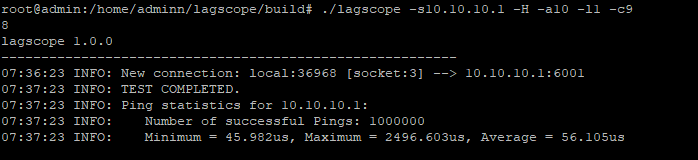


# lagscope

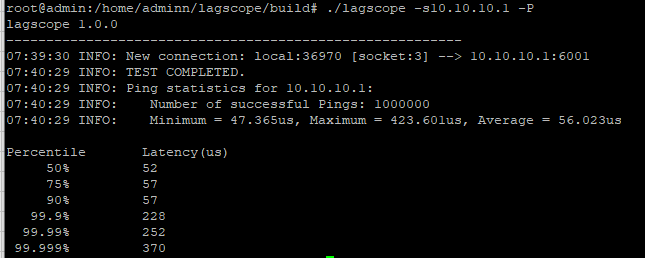
A Linux/Windows tool to measure the network transport layer latency.

* Support network transport layer latency measurement (round-trip latency).
* Support specifying ping message size, and ping interval.
* Support two test modes: test-duration mode and ping-iteration mode.
* Support histogram reporting and percentile reporting.
* Support interop test with Windows latte.exe.
* Support running natively on both Linux and Windows Operating Systems.
* Python3 api
* UDP is not supported.

Test1



Test2:

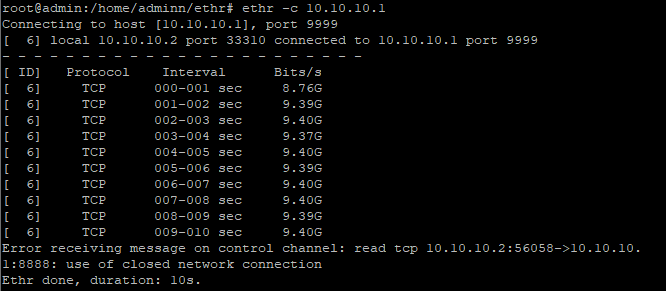


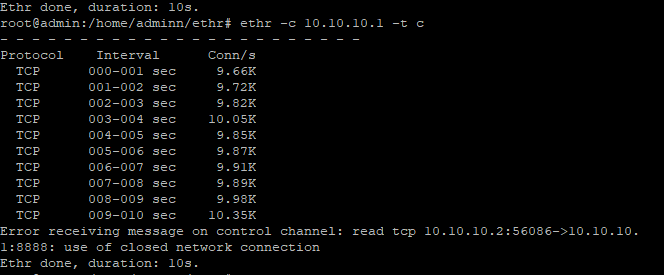
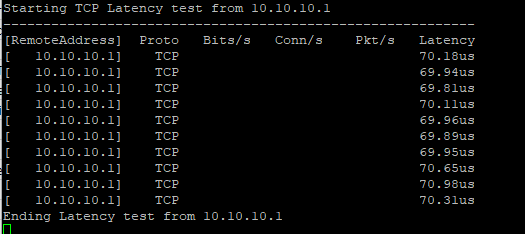
# Ethr

Written with go

Ethr is a cross platform network performance measurement tool written in golang. The goal of this project is to provide a native tool for comprehensive network performance measurements of bandwidth, connections/s, packets/s, latency, loss & jitter, across multiple protocols such as TCP, UDP, HTTP, HTTPS, and across multiple platforms such as Windows, Linux and other Unix systems. Ethr takes inspiration from existing open source network performance tools and builds upon those ideas. For Bandwidth measurement, it is similar to iPerf3, for TCP & UDP traffic. iPerf3 has many more options for doing such as throttled testing, richer feature set, while Ethr has support for multiple threads, that allows it to scale to 1024 or even higher number of connections, multiple clients communication to a single server etc. For latency measurements, it is similar to latte on Windows or sockperf on Linux.

Ethr provides more test measurements as compared to other tools, e.g. it provides measurements for bandwidth, connections/s, packets/s, latency, and TCP connection setup latency, all in a single tool.





# NTTTCP

It is accurate

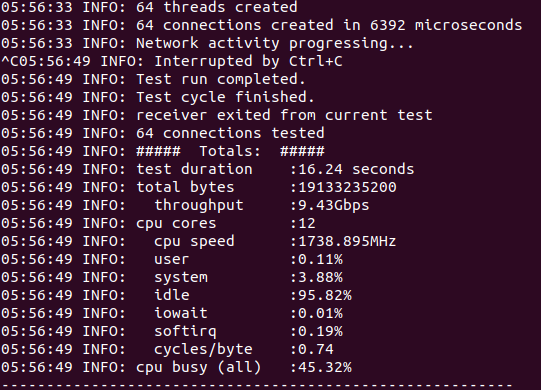
Code is written in C.

A multiple-threaded Linux network throughput benchmark tool.

## Features

* Multiple threads to send/receive data ('-P', '-n', and '-l'). By default, Receiver ('-r') uses 16 threads and Sender ('-s') uses 64 threads to exchange data.
* Support cpu affinity ('-m').
* Support running in background (daemon, '-D').
* Support Sender and Receiver sync mode by default. Use "-N" (no\_sync) to disable the sync.
* Support testing with multiple clients mode (use '-M' on Receiver, and '-L' on the last Sender).
* Support select() by default, and epoll() (use '-e' on Receiver).
* Support both TCP (by default), and UDP ('-u') tests.
* Support pin TCP server or client port (use '-p' on Receiver or '-f' on Sender).
* Support test Warmup ('-W') and Cooldown ('-C').
* Support reporting TCP retransmit ('--show-tcp-retrans').
* Support reporting number of packets ('--show-nic-packets') and number of interrupts ('--show-dev-interrupts').
* Support bandwidth limit ('-B' or '--fq-rate-limit').
* Support writing log into XML file ('-x').
* Support capturing console log to file ('-O').

<https://github.com/microsoft/ntttcp-for-linux>



# Speedtest

A TCP utility to test the available bandwidth of the network.

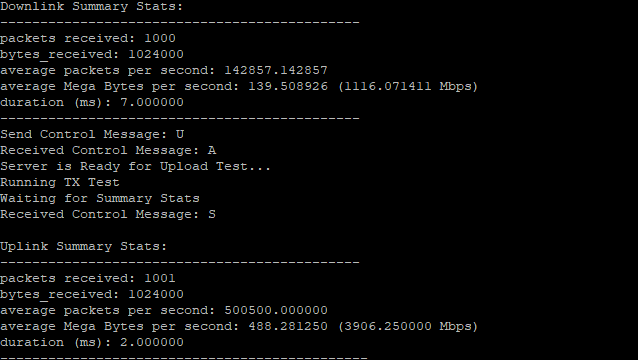
This utility consists of two binaries server and client. The server can run on any machine, the client shoule know the ip address or the domain name of the server to connect and start the test. The tests will check both the uplink and downlink bandwidth for the client.

Code is written in C.

**./server**

**/client --server-ip <server ip or domain name> --server-port <port no>**

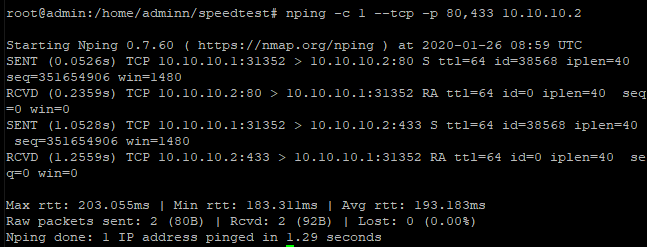
The throughput is calculated on receiver side (so for uplink it is calculated by server and sent using a control message to the client, for downling client will calculate and display the data).



<https://github.com/smihir/speedtest>

It is not accurate.

# Nping



nping: Network packet generation tool / ping utility

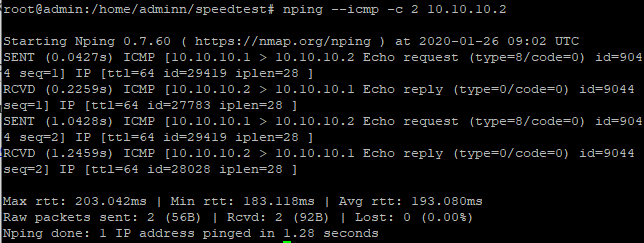
example: nping -c 1 --tcp -p 80,433 37.152.189.242

Nping is an open-source tool for network packet generation, response analysis and response time measurement. Nping allows users to generate network packets of a wide range of protocols, letting them tune virtually any field of the protocol headers. While Nping can be used as a simple ping utility to detect active hosts, it can also be used as a raw packet generator for network stack stress tests, ARP poisoning, Denial of Service attacks, route tracing, and other purposes.

Additionally, Nping offers a special mode of operation called the "Echo Mode", that lets users see how the generated probes change in transit, revealing the differences between the transmitted packets and the packets received at the other end. See section "Echo Mode" for details.

The output from Nping is a list of the packets that are being sent and received. The level of detail depends on the options used.

A typical Nping execution is shown in Example 1. The only Nping arguments used in this example are **-c**, to specify the number of times to target each host, **--tcp** to specify TCP Probe Mode, **-p 80,433** to specify the target ports; and then the two target hostnames.



<https://github.com/nmap/nmap/tree/master/nping>

# Goben

* upport for TCP, UDP, TLS.
* Can limit maximum bandwidth.
* Written in [Go](https://golang.org/). Single executable file. No runtime dependency.
* Simple usage: start the server then launch the client pointing to server's address.
* Spawns multiple concurrent lightweight goroutines to handle multiple parallel traffic streams.
* Can save test results as PNG chart.
* Can export test results as YAML or CSV.

