In the name of God

1. **Iperf 2,3 (test+)**

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

**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>

1. **MTR(test +)**

Once you run **mtr**, it probes the network connection between the local system and a remote host that you have specified. It first establishes the address of each network hop (bridges, routers and gateways etc.) between the hosts, it then **pings** (sends a sequence **ICMP ECHO** requests to) each one to determine the quality of the link to each machine.

You can use TCP SYN packets or UDP datagrams instead of the default ICMP ECHO requests as shown.

Sudo apt install mtr

Keys: Help Display mode Restart statistics Order of fields quit

Packets Pings

Host Loss% Snt Last Avg Best Wrst StDev

1. 37.152.176.2 2.9% 240 0.7 0.8 0.6 4.6 0.4

2. 172.28.1.178 0.0% 240 3.4 2.2 1.3 12.4 1.8

3. 172.19.32.30 0.4% 240 1.5 1.9 1.2 12.8 1.4

4. 172.19.32.1 0.0% 240 2.2 2.3 1.3 24.2 2.4

5. 10.201.181.78 0.8% 240 3.4 3.3 1.9 15.6 1.7

6. 10.101.101.30 0.0% 239 5.1 4.7 2.4 17.3 2.5

7. 222.mobinnet.net 7.5% 239 1.9 2.4 1.4 18.8 1.7

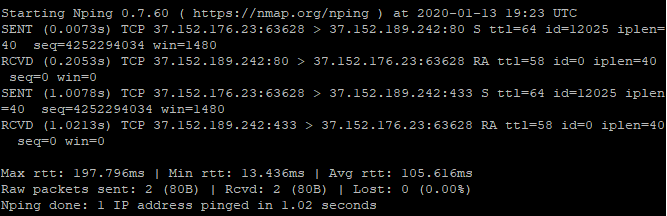
8. 37.152.189.242 0.4% 239 2.8 2.6 1.6 12.3 1.4

1. **Nping(test+)**

Code is written in C.

nping: Network packet generation tool / ping utility

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

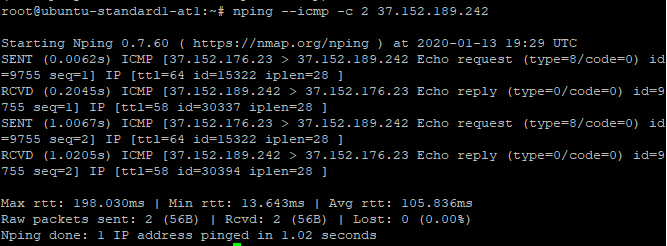


Nping is an open-source tool for network packet generation, response analysis and response time measurement. Nping allows users to generate network packets of a wide range of protocols, letting them tune virtually any field of the protocol headers. While Nping can be used as a simple ping utility to 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>

1. **Ntttcp (test +)**

It is accurate

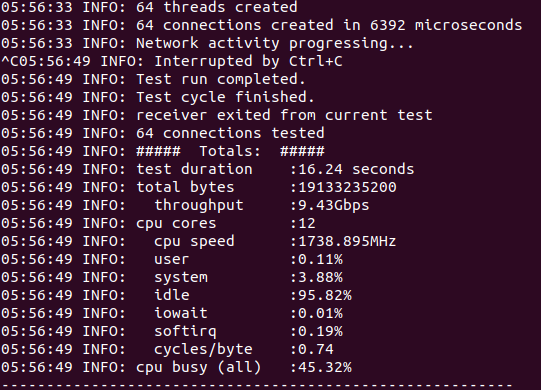
Code is written in C.

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>



1. **Speedtest (test+)**

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

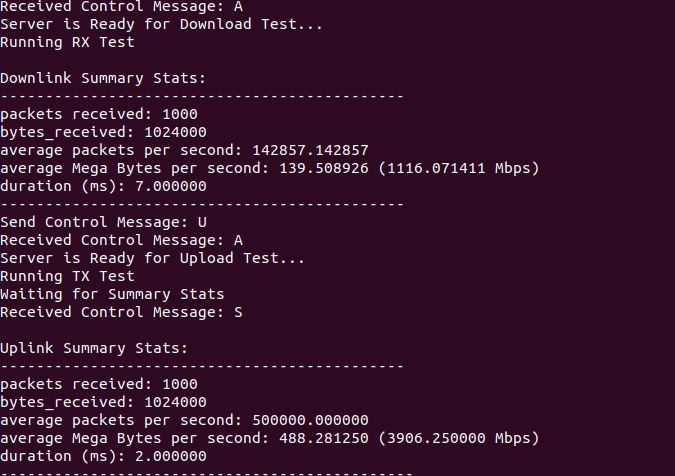
This utility consists of two binaries server and client. The server can run on any machine, the client shoule know the ip address or the domain name of the server to connect and start the test. The tests will check both the uplink and downlink bandwidth for the client.

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.

1. **NetCat (test+)**

**Port Scanning**

**Netcat** can be used for port scanning: to know [which ports are open](https://www.tecmint.com/find-listening-ports-linux/) and running services on a target machine. It can scan a single or multiple or a range of open ports.

**Transfer Files Between Linux Servers**

**Netcat** allows you to [transfer files between two Linux computers or servers](https://www.tecmint.com/transfer-files-between-two-linux-machines/) and both these systems must have **nc** installed.

**Create a Command Line Chat Server**

You can also use **Netcat** to create a simple [command-line messaging server](https://www.tecmint.com/linux-commandline-chat-server-and-remove-unwanted-packages/) instantly. As in the previous usage example, **nc** must be installed on both systems used for the chat room.

Create a Basic Web Server

Wit the **-l** option of **nc command** used to create a basic, insecure web server to serve static web files for learning purposes. To demonstrate this, create a **.html** file as shown.

**Troubleshoot Linux Server Connection**

Another useful usage of **Netcat** is to [troubleshoot server connection](https://www.tecmint.com/linux-networking-commands/) issues. Here, you can use **Netcat** to verify what data a server is sending in response to commands issued by the client.

Find a Service Running on Port

You can also use **Netcat** to obtain port banners. In this case, it will tell you what service is running behind a certain port. For example to know what type of service is running behind port **22** on a specific server, run the following command (replace **192.168.56.110** with the target server’s IP address). The **-n** flag means to disable DNS or service lookups.

**Create a Stream Sockets**

**Netcat** also supports creation of UNIX-domain stream sockets. The following command will create and listen on a UNIX-domain stream socket.

Create a Backdoor

You can as well run **Netcat** as a backdoor. However, this calls for more work. If **Netcat** is installed on a target server, you can use it to create a backdoor, to get a remote command prompt.

To act a backdoor you need **Netcat** to listen on a chosen port (e.g port **3001**) on the target server and then you can connect to this port from your machine as follows.

This is the command intended to run on the remote server where the **-d** option disables reading from stdin, and **-e** specifies the command to run on the target system.

Last but not least, **Netcat** can be used as a proxy for different services/protocols including HTTP, SSH, and many more. For more information, see its man page.

1. **Hping3(test+)**

**hping3** is another tool used for scan network. it is available in kali linux by default it is one of **DOS attack software**, ddos stand for **distributed denial of service attack**. you can launch and **stop dos attack**, whenever you want. In this illustration **hping3** will act like an ordinary ping utility, sending ICMP-reverberation und getting ICMP-reply.



**Traceroute using Hping3:**

This illustration is like popular utilities like tracert (windows) or traceroute (linux) who utilizes ICMP packets expanding each time in 1 its TTL value.

HPing is a tool which you can assemble custom ICMP, UDP or TCP packets. These packets can then be sent and you can see replies (if any). The tool is very useful when it comes to pretesting and network analysis. Using the settings in the tool, tracerouting, pinging and probing hosts behind firewalls is fairly easy. Setting it up properly, you’re able to map out a firewall and its rules. Worth mentioning is that people today tend to use this (actually pretty old tool) as something to stress test networks/hosts with. (DoS attacks). That’s not the only intention of this tool, but people seem to miss that. As we went over in the video, we quickly went over the different tags that could be used. Explaining some of the most common ones.

1. **Tcpkali (test-)**

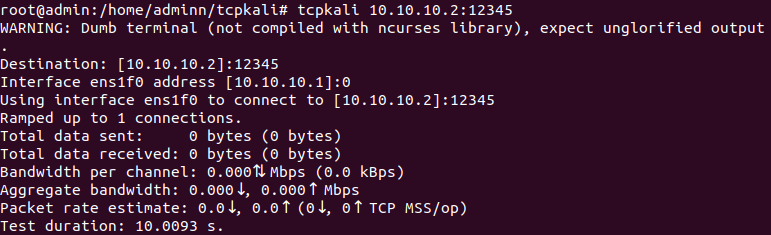
tcpkali is a high performance TCP and WebSocket load generator and sink.

**Features**

* Opens millions of connections from a single host by using available interface aliases.
* Efficient multi-core operation (--workers); utilizes all available cores by default.
* Allows opening massive number of connections (--connections)
* Allows limiting an upstream and downstream of a single connection throughput (--channel-bandwidth-downstream, --channel-bandwidth-upstream or --message-rate)
* Allows specifying the first and subsequent messages (--message, --first-message).
* Measures response latency percentiles using [HdrHistogram](https://github.com/HdrHistogram) (--latency-marker)
* Sends stats to StatsD/DataDog (--statsd)

<https://github.com/akumuli/tcpkali>

The result of test is not clear.



Example of tests:

Connect to a local web server and do nothing

Connect to a local echo server and hammer it

Open x connections to two remote servers

Open x connections to itself and do nothing

Open a connection to itself and send lots of cookies

Listen for incoming connections and throw away data for x hours

Open connection to the local WebSocket server, send hello, and wait

1. **Warp17**

<https://github.com/Juniper/warp17>

Python and Perl API

WARP17, The Stateful Traffic Generator for L1-L7 is a lightweight solution for generating high volumes of session based traffic with very high setup rates. WARP17 currently focuses on L5-L7 application traffic (e.g., HTTP) running on top of TCP as this kind of traffic requires a complete TCP implementation. Nevertheless, WARP17 also supports application traffic running on top of UDP.

Developing network components or services usually requires expensive proprietary solutions for validating the implemented functionalities and scalability or performance requirements. WARP17 is a platform agnostic tool based on [DPDK](http://dpdk.org/) which:

* allows extremely fast generation and sustaining of stateful sessions
* offers configurable TCP/UDP infrastructure which can be used for generating high connection setup and data rates for application traffic
* is [Linux](https://kernel.org/) based so all the openly available tools can be integrated by the users of WARP17.

The WARP17 TCP/IP implementation runs completely in user-space thus avoiding the additional latency in the kernel stack. From a hardware perspective, WARP17 will be able to run on all the platforms that are supported by DPDK.