# iperf3 Documentation

*Release 0.1.10*

**Mathijs Mortimer**

**Mar 27, 2018**

# Contents


**1 Installation** **3**
- 1.1 iperf3 utility . . . 3
- 1.2 iperf3 python wrapper . . . 3

**2 Examples** **5**
- 2.1 Client . . . 5
- 2.2 Server . . . 6

**3 Modules** **9**
- 3.1 iperf3 . . . 9
  - 3.1.1 Client . . . 9
  - 3.1.2 Server . . . 10
  - 3.1.3 TestResult . . . 11
  - 3.1.4 IPerf3 . . . 12

**Python Module Index** **15**

Release v0.1.10.

iPerf3 is a tool for active measurements of the maximum achievable bandwidth on IP networks. More information on the iPerf3 utility can be found on their official website

The python iperf3 module is a wrapper around the iperf3 utility. It utilises the API libiperf that comes with the default installation. It allows you to interact with the utility in a nice and pythonic way.

**warning** This module is not compatible with the original iperf/iperf2 utility which is no longer under active development

CHAPTER 1

# Installation

To be able to utilise the python wrapper around iperf3 you will need to have the libiperf.so.0 shared library installed. Luckily this comes with the standard iperf3 build.

## 1.1 iperf3 utility

Preferably get the latest build from the iperf3 official website

Otherwise try your OS package manager:

- Ubuntu:

```
sudo apt-get install iperf3
```

- CentOS/RedHat

```
sudo yum install iperf3
```

## 1.2 iperf3 python wrapper

The preferred installation method is through PyPi (aka pip install)

```
pip install iperf3
```

If pip is unavailable for any reason you can also manually install from github:

```
git clone https://github.com/thiezn/iperf3-python.git
cd iperf3-python
python3 setup.py test  # (optional) testing through py.test and/or tox
python3 setup.py install
```

CHAPTER 2

# Examples

Check the examples/ folder for a few ready to go python scripts.

## 2.1 Client

**Example 1**

This example sets up a client connection to a running server on 10.10.10.10:6969. When the test finalises the results are returned. This example shows all currently available options for a `Client`

```
>>> import iperf3
```

```
>>> client = iperf3.Client()
>>> client.duration = 1
>>> client.bind_address = '10.0.0.1'
>>> client.server_hostname = '10.10.10.10'
>>> client.port = 6969
>>> client.blksize = 1234
>>> client.num_streams = 10
>>> client.zerocopy = True
>>> client.verbose = False
>>> client.reverse = True
>>> client.run()
{'start': {'test_start': {...
```

**Example 2**

This example shows how you can output the client test results to screen, just like the iperf3 application itself does. Note it does NOT return a `TestResult` instance.

```
>>> import iperf3
```

```
>>> client = iperf3.Client()
>>> client.server_hostname = '10.10.10.10'
>>> client.port = 6969
>>> client.json_output = False
>>> result = client.run()
Time: Mon, 15 May 2017 18:20:01 GMT
Connecting to host 10.10.10.10, port 6969
[  8] local 127.0.0.1 port 35670 connected to 127.0.0.1 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 1␣
↪second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  8]   0.00-1.00   sec  3.96 GBytes  34.0 Gbits/sec    0   3.18 MByt...
```

```
>>> result
None
```

**Example 3**

Here is an example of running a UDP test. Please read the official documentation on UDP testing as there can be a few catches.

```
#!/usr/bin/env python3

import iperf3

client = iperf3.Client()
client.duration = 1
client.server_hostname = '127.0.0.1'
client.port = 5201
client.protocol = 'udp'


print('Connecting to {0}:{1}'.format(client.server_hostname, client.port))
result = client.run()

if result.error:
    print(result.error)
else:
    print('')
    print('Test completed:')
    print('  started at         {0}'.format(result.time))
    print('  bytes transmitted  {0}'.format(result.bytes))
    print('  jitter (ms)        {0}'.format(result.jitter_ms))
    print('  avg cpu load       {0}%\n'.format(result.local_cpu_total))

    print('Average transmitted data in all sorts of networky formats:')
    print('  bits per second      (bps)   {0}'.format(result.bps))
    print('  Kilobits per second  (kbps)  {0}'.format(result.kbps))
    print('  Megabits per second  (Mbps)  {0}'.format(result.Mbps))
    print('  KiloBytes per second (kB/s)  {0}'.format(result.kB_s))
    print('  MegaBytes per second (MB/s)  {0}'.format(result.MB_s))
```

## 2.2 Server

**Example 1**

This example runs an iperf3 server on 10.10.10.10:6969 and prints out the test results. After each test `server.run()` finishes and produces the test results. This example shows all currently available options for a *Server*

```
>>> import iperf3
```

```
>>> server = iperf3.Server()
>>> server.bind_address = '10.10.10.10'
>>> server.port = 6969
>>> server.verbose = False
>>> while True:
...     server.run()
...
{'start': {'test_start': {...
```

CHAPTER 3

# Modules

## 3.1 iperf3

Python wrapper for the iperf3 libiperf.so.0 library. The module consists of two classes, `Client` and `Server`, that inherit from the base class `IPerf3`. They provide a nice (if i say so myself) and pythonic way to interact with the iperf3 utility.

At the moment the module redirects stdout and stderr to a pipe and returns the received data back after each `client.run()` or `server.run()` call. In later releases there will be an option to toggle this on or off.

A user should never have to utilise the `IPerf3` class directly, this class provides common settings for the `Client` and `Server` classes.

To get started quickly see the *Examples* page.

### 3.1.1 Client

**class** iperf3.**Client**(**args*, ***kwargs*)

An iperf3 client connection.

This opens up a connection to a running iperf3 server

Basic Usage:

```
>>> import iperf3

>>> client = iperf3.Client()
>>> client.duration = 1
>>> client.server_hostname = '127.0.0.1'
>>> client.port = 5201
>>> client.run()
{'intervals': [{'sum': {...
```

**bandwidth**

Target bandwidth in bits/sec

**blksize**
The test blksize.

**bulksize**
The test bulksize.

Deprecated argument, use blksize instead to ensure consistency with iperf3 C libary

**duration**
The test duration in seconds.

**num_streams**
The number of streams to use.

**protocol**
The iperf3 instance protocol

valid protocols are 'tcp' and 'udp'

**Return type** str

**reverse**
Toggles direction of test

**Return type** bool

**run**()
Run the current test client.

**Return type** instance of *TestResult*

**server_hostname**
The server hostname to connect to.

Accepts DNS entries or IP addresses.

**Return type** string

**zerocopy**
Toggle zerocopy.

Use the sendfile() system call for "Zero Copy" mode. This uses much less CPU. This is not supported on all systems.

**Note** there isn't a hook in the libiperf library for getting the current configured value. Relying on zerocopy.setter function

**Return type** bool

### 3.1.2 Server

**class** iperf3.**Server**(**args*, ***kwargs*)
An iperf3 server connection.

This starts an iperf3 server session. The server terminates after each succesful client connection so it might be useful to run Server.run() in a loop.

The C function iperf_run_server is called in a seperate thread to make sure KeyboardInterrupt(aka ctrl+c) can still be captured

Basic Usage:

```
>>> import iperf3

>>> server = iperf3.Server()
>>> server.run()
{'start': {...
```

**run**()

Run the iperf3 server instance.

**Return type** instance of *TestResult*

### 3.1.3 TestResult

**class** iperf3.**TestResult**(*result*)

Class containing iperf3 test results.

**Parameters**

- **text** – The raw result from libiperf as text
- **json** – The raw result from libiperf asjson/dict
- **error** – Error captured during test, None if all ok
- **time** – Start time
- **timesecs** – Start time in seconds
- **system_info** – System info
- **version** – Iperf Version
- **local_host** – Local host ip
- **local_port** – Local port number
- **remote_host** – Remote host ip
- **remote_port** – Remote port number
- **reverse** – Test ran in reverse direction
- **protocol** – 'TCP' or 'UDP'
- **num_streams** – Number of test streams
- **blksize** –
- **omit** –
- **duration** – Test duration in seconds
- **local_cpu_total** – The local total CPU load
- **local_cpu_user** – The local user CPU load
- **local_cpu_system** – The local system CPU load
- **remote_cpu_total** – The remote total CPU load
- **remote_cpu_user** – The remote user CPU load
- **remote_cpu_system** – The remote system CPU load

TCP test specific

**Parameters**

- **tcp_mss_default** –
- **retransmits** – amount of retransmits (Only returned from client)
- **sent_bytes** – Sent bytes
- **sent_bps** – Sent bits per second
- **sent_kbps** – sent kilobits per second
- **sent_Mbps** – Sent Megabits per second
- **sent_kB_s** – Sent kiloBytes per second
- **sent_MB_s** – Sent MegaBytes per second
- **received_bytes** – Received bytes
- **received_bps** – Received bits per second
- **received_kbps** – Received kilobits per second
- **received_Mbps** – Received Megabits per second
- **received_kB_s** – Received kiloBytes per second
- **received_MB_s** – Received MegaBytes per second

UDP test specific

**Parameters**

- **bytes** –
- **bps** –
- **jitter_ms** –
- **kbps** –
- **Mbps** –
- **kB_s** –
- **MB_s** –
- **packets** –
- **lost_packets** –
- **lost_percent** –
- **seconds** –

### 3.1.4 IPerf3

**class** `iperf3.`**`IPerf3`**(*role*, *verbose=True*, *lib_name=None*)

The base class used by both the iperf3 `Server` and `Client`

**Note:** You should not use this class directly

**bind_address**
The bind address the iperf3 instance will listen on

use * to listen on all available IPs :rtype: string

**defaults**()
Set/reset iperf test defaults.

**iperf_version**
Returns the version of the libiperf library

**Return type** string

**json_output**
Toggles json output of libiperf

Turning this off will output the iperf3 instance results to stdout/stderr

**Return type** bool

**port**
The port the iperf3 server is listening on

**role**
The iperf3 instance role

valid roles are 'c'=client and 's'=server

**Return type** 'c' or 's'

**run**()
Runs the iperf3 instance.

This function has to be instantiated by the Client and Server instances

**Return type** NotImplementedError

**verbose**
Toggles verbose output for the iperf3 instance

**Return type** bool

# Python Module Index

## i

iperf3, 9

# Index


## B

bandwidth (iperf3.Client attribute), 9
bind_address (iperf3.IPerf3 attribute), 12
blksize (iperf3.Client attribute), 9
bulksize (iperf3.Client attribute), 10

## C

Client (class in iperf3), 9

## D

defaults() (iperf3.IPerf3 method), 13
duration (iperf3.Client attribute), 10

## I

IPerf3 (class in iperf3), 12
iperf3 (module), 9
iperf_version (iperf3.IPerf3 attribute), 13

## J

json_output (iperf3.IPerf3 attribute), 13

## N

num_streams (iperf3.Client attribute), 10

## P

port (iperf3.IPerf3 attribute), 13
protocol (iperf3.Client attribute), 10

## R

reverse (iperf3.Client attribute), 10
role (iperf3.IPerf3 attribute), 13
run() (iperf3.Client method), 10
run() (iperf3.IPerf3 method), 13
run() (iperf3.Server method), 11

## S

Server (class in iperf3), 10
server_hostname (iperf3.Client attribute), 10

## T

TestResult (class in iperf3), 11

## V

verbose (iperf3.IPerf3 attribute), 13

## Z

zerocopy (iperf3.Client attribute), 10