Throughput Measurement

# Introduction

This document describes how to characterize and analyze the Wi-Fi throughput on Talaria TWO using iPerf3 application accompanied with the SDK.

The iPerf application is a tool built on a client/server model which is used for measuring TCP and UDP bandwidth performance. Throughput is measured between a test PC and Talaria TWO, each running an iPerf3 application.

Various test configurations are described, including download and upload throughput tests, TCP and UDP throughput tests, and tests of a specific duration, data size, or data rate.

# iPerf3 features

Following are a few notable features of iPerf3 which makes it more user friendly than any of the previous versions of iPerf.

1. iPerf3 supports Dynamic Server, which allows Client/Server parameter exchange.

This implies that the Server can get different test parameters from Client-side command line without having to restart the Server after one test completes. This is achieved by having a separate control TCP Socket to manage messages between Client and Server.

1. By default, the Server receives the data and the Client sends the data.

iPerf3 also supports Reverse Mode where the Server sends the data and the client receives it.

1. iPerf3 has a feature of Client/Server Results Exchange where the Client and Server can see each other's results.

These features enable us to start a Server on the DUT and all the options of test parameters can be passed from the Client-side command line remotely running on the PC.

Once the test completes, the next test can also be triggered from the Client-side with different parameters.

For example, both UDP and TCP tests can be initiated from remote PC one by one, without the need for restarting the DUT in a different mode.

Both downlink and uplink tests on the DUT can be triggered from the Client-side command line using default mode and reverse mode, respectively.

The result from the DUT running Server is seen on Client-side remote console as well.

# Prerequisites

To run the tests described in this document, a test PC is required. The PC needs to be accessible from the access point to which Talaria TWO will connect to.

Both the Talaria TWO and the test PC run their own versions of iPerf3. Talaria TWO iPerf3 application is available in the SDK as described in the section 6.

The latest iPerf3 application is not officially supported for Windows OS PC. This document describes the use of the iPerf3 application on an Ubuntu PC. However, the use of iPerf3 on other supporting operating systems will be similar.

Command to install latest version of iPerf3 on Ubuntu is as follows:

|  |
| --- |
| sudo apt-get install iperf3 |

**Note**: iPerf3 version 3.6-2 is used for the purpose of description in this document.

# Throughput testing with iPerf3 on Talaria TWO

The iPerf3 application available in the SDK at path apps/iperf3/iperf3.elf, implements iPerf3 functionality on Talaria TWO.

The iPerf3 application running on Talaria TWO always operates in the Server mode, listening on port 5001 and waiting to accept the Client’s connection before it initiates the bandwidth test.

Multiple other configurations for running throughput test are passed via Client-side command line.

This section describes how the application can be used to perform various throughput tests.

## Running iPerf3 on Talaria TWO and Connecting to the Access Point

Program iperf3.elf (sdk\_x.y\apps\iperf3\bin) using the Download Tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down
   2. ELF Input: Load the iperf3.elf by clicking on Select ELF File.
   3. AP Options: Provide the SSID and Passphrase under AP Options to connect to an Access Point.
   4. Boot arguments: Pass the following boot arguments:

**Note**: Boot Arguments are optional and enables additional information printouts on Talaria TWO’s end.

|  |
| --- |
| iperf3.extra\_info=1 |

* 1. Programming: Prog RAM or Prog Flash as per requirement.

For more details on using the Download tool, refer to the document: UG\_Download\_Tool.pdf (path: *sdk\_x.y/pc\_tools/Download\_Tool/doc*).

**Note**: x & y refer to SDK release version. For example: sdk\_x.y/pc\_tools/Download\_Tool/doc.

For example, parameters received from client over TCP control socket at the beginning of the test and Server/Client results at the end of the test are printed on Talaria TWO if this boot argument is enabled.

Console Output:

**Note**: The following console output is from SDK 2.3 release and is applicable to the current release as well.

|  |
| --- |
| UART:NWWWWWWAEBuild $Id: git-f92bee540 $  ssid=ACT102571068294 passphrase=43083191 iperf3.extra\_info=1  addr e0:69:3a:00:2c:3e  [10.403,370] CONNECT:00:5f:67:cd:c5:a6 Channel:6 rssi:-16 dBm  [11.190,449] MYIP 192.168.0.105  [11.190,612] IPv6 [fe80::e269:3aff:fe00:2c3e]-link  IPerf3 server @ 192.168.0.105  ----------------------------------------  Iperf3 TCP/UDP server listening on 5201  ----------------------------------------  Accepted connection from 192.168.0.101 port 1060  [172.166,310] iperf3: param = {"tcp":true,"omit":0,"time":20,"parallel":1,"len":131072,"pacing\_timer":1000,"client\_version":"3.7"}  [ 1] local 192.168.0.105 port 5201 connected to 192.168.0.101 port 1065  [192.284,574] iperf3: client\_result = {"cpu\_util\_total":0.46521064270562279,"cpu\_util\_user":0.060944126128725948,"cpu\_util\_system":0.40426651657689677,"sender\_has\_retransmits":1,"congestion\_used":"cubic","streams":[{"id":1,"bytes":28204280,"retransmits":0,"jitter":0,"errors":0,"packets":0,"start\_time":0,"end\_time":20.00013}]}  ------------------------------  [ ID] Interval Transfer Bitrate  [ 1] iperf3[S-RX-tcp]: 0.0-20 sec 26.3 MBytes 11.0 Mbits/sec  User: 8567617 (42%)  IRQ: 1015533 (5%)  Idle: 10442621 (52%)  [192.286,398] iperf3: server\_result = {"cpu\_util\_total": 47,"cpu\_util\_user": 42,"cpu\_util\_system": 5,"sender\_has\_retransmits": -1,"congestion\_used": "cubic","streams": [{"id":1,"bytes":27607140,"retransmits":0,"jitter":0,"errors":0,"packets":0,"start\_time":0,"end\_time":20.000100}]}  ----------------------------------------  Iperf3 TCP/UDP server listening on 5201  ---------------------------------------- |

Take note of the IP address of the iPerf3 server displayed in the console.

Once Talaria TWO has booted and the iPerf3 has been started, the Ubuntu PC can act as an iPerf3 Client and can connect to the Talaria TWO and run a throughput test.

Console output from Talaria TWO when iPerf3 application is booted is as follows:

**Note**: The following console outputs are from SDK 2.3 release and is applicable to the current release as well.

|  |
| --- |
| Accepted connection from 192.168.0.101 port 1050  [58.745,191] iperf3: param = {"tcp":true,"omit":0,"time":20,"parallel":1,"len":131072,"pacing\_timer":1000,"client\_version":"3.7"}  [ 1] local 192.168.0.105 port 5201 connected to 192.168.0.101 port 1052  [78.820,085] iperf3: client\_result = {"cpu\_util\_total":0.32061508034327585,"cpu\_util\_user":0.025213145835174724,"cpu\_util\_system":0.29539699946291903,"sender\_has\_retransmits":1,"congestion\_used":"cubic","streams":[{"id":1,"bytes":32572600,"retransmits":0,"jitter":0,"errors":0,"packets":0,"start\_time":0,"end\_time":20.000591}]}  ------------------------------  [ ID] Interval Transfer Bitrate  [ 1] iperf3[S-RX-tcp]: 0.0-20 sec 30.7 MBytes 12.8 Mbits/sec  User: 9950311 (49%)  IRQ: 1192082 (5%)  Idle: 8910179 (44%)  [78.821,837] iperf3: server\_result = {"cpu\_util\_total": 54,"cpu\_util\_user": 49,"cpu\_util\_system": 5,"sender\_has\_retransmits": -1,"congestion\_used": "cubic","streams": [{"id":1,"bytes":32201760,"retransmits":0,"jitter":0,"errors":0,"packets":0,"start\_time":0,"end\_time":20.000629}]} |

Following sections describe different commands to be given at iPerf3 Client-side to perform various throughput tests.

## Time to Run the Test

The iPerf3 Client on the remote PC can be started with the -t option to specify test duration.

It specifies the time to run the test in seconds. If this option is not specified, the iPerf 3.6 on Ubuntu runs the test for its default setting of 10 seconds. For most of the examples we will use option -t 20.

## Talaria TWO Downlink TCP Throughput

This is accomplished by running the following command on the PC iPerf3 Client, using the Talaria TWO iPerf3 Server’s IP address:

|  |
| --- |
| iperf3 -c 192.168.43.60 -t 20 |

The test measures throughput from Client (PC) to Server (T2) over TCP, which is ingress /downlink for Talaria TWO.

The individual results are displayed in the Ubuntu PC iPerf3 shell as well as in the Talaria TWO console. Results from Talaria TWO are shown in Ubuntu PC iPerf3 shell as well.

The default length of the buffer to read or write is 128 KB (131072 B) for TCP.

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -t 20  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 36334 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Retr Cwnd  [ 5] 0.00-1.00 sec 1.22 MBytes 10.2 Mbits/sec 0 68.4 KBytes  [ 5] 1.00-2.00 sec 1.01 MBytes 8.48 Mbits/sec 0 69.9 KBytes  [ 5] 2.00-3.00 sec 1.23 MBytes 10.3 Mbits/sec 0 68.4 KBytes  [ 5] 3.00-4.00 sec 1.35 MBytes 11.3 Mbits/sec 0 68.4 KBytes  [ 5] 4.00-5.00 sec 1.96 MBytes 16.4 Mbits/sec 0 68.4 KBytes  [ 5] 5.00-6.00 sec 1.90 MBytes 15.9 Mbits/sec 0 68.4 KBytes  [ 5] 6.00-7.00 sec 1.56 MBytes 13.1 Mbits/sec 0 69.9 KBytes  [ 5] 7.00-8.00 sec 1.07 MBytes 9.00 Mbits/sec 0 69.9 KBytes  [ 5] 8.00-9.00 sec 1.90 MBytes 15.9 Mbits/sec 0 69.9 KBytes  [ 5] 9.00-10.00 sec 1.90 MBytes 15.9 Mbits/sec 0 69.9 KBytes  [ 5] 10.00-11.00 sec 1.59 MBytes 13.4 Mbits/sec 0 69.9 KBytes  [ 5] 11.00-12.00 sec 1.84 MBytes 15.4 Mbits/sec 0 69.9 KBytes  [ 5] 12.00-13.00 sec 1.75 MBytes 14.6 Mbits/sec 0 69.9 KBytes  [ 5] 13.00-14.00 sec 1.44 MBytes 12.1 Mbits/sec 0 69.9 KBytes  [ 5] 14.00-15.00 sec 1.65 MBytes 13.9 Mbits/sec 0 69.9 KBytes  [ 5] 15.00-16.00 sec 1.90 MBytes 15.9 Mbits/sec 0 69.9 KBytes  [ 5] 16.00-17.00 sec 1.38 MBytes 11.6 Mbits/sec 0 69.9 KBytes  [ 5] 17.00-18.00 sec 1.19 MBytes 10.0 Mbits/sec 0 69.9 KBytes  [ 5] 18.00-19.00 sec 1.44 MBytes 12.1 Mbits/sec 0 69.9 KBytes  [ 5] 19.00-20.00 sec 1.26 MBytes 10.5 Mbits/sec 0 68.4 KBytes  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Retr  [ 5] 0.00-20.00 sec 30.5 MBytes 12.8 Mbits/sec 0 sender  [ 5] 0.00-20.00 sec 30.2 MBytes 12.6 Mbits/sec receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

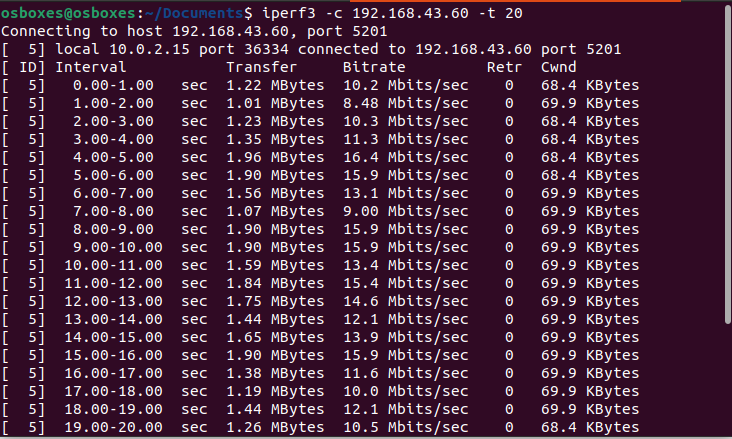


Figure : Talaria TWO Downlink TCP Throughput - Console Output

## Talaria TWO Uplink TCP Throughput

This is accomplished by using Reverse Mode option -R when running iPerf3 Client using the Talaria TWO iPerf3 Server’s IP address:

|  |
| --- |
| iperf3 -c 192.168.43.60 -t 20 -R |

As its run in Reverse Mode, the test measures throughput from Server (Talaria TWO) to Client (PC) over TCP, which is egress /uplink for the Talaria TWO.

The individual results are displayed in the Ubuntu PC iperf3 shell as well as in the Talaria TWO console. Results from Talaria TWO are shown in Ubuntu PC iperf3 shell as well.

The default length of the buffer to read or write is 128 KB (131072 B) for TCP.

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -t 20 -R  Connecting to host 192.168.43.60, port 5201  Reverse mode, remote host 192.168.43.60 is sending  [ 5] local 10.0.2.15 port 36366 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate  [ 5] 0.00-1.00 sec 1.56 MBytes 13.1 Mbits/sec  [ 5] 1.00-2.00 sec 515 KBytes 4.21 Mbits/sec  [ 5] 2.00-3.00 sec 187 KBytes 1.53 Mbits/sec  [ 5] 3.00-4.00 sec 1.72 MBytes 14.5 Mbits/sec  [ 5] 4.00-5.00 sec 2.11 MBytes 17.7 Mbits/sec  [ 5] 5.00-6.00 sec 2.07 MBytes 17.4 Mbits/sec  [ 5] 6.00-7.00 sec 2.10 MBytes 17.6 Mbits/sec  [ 5] 7.00-8.00 sec 1.78 MBytes 14.9 Mbits/sec  [ 5] 8.00-9.00 sec 1.68 MBytes 14.1 Mbits/sec  [ 5] 9.00-10.00 sec 1.38 MBytes 11.6 Mbits/sec  [ 5] 10.00-11.00 sec 0.00 Bytes 0.00 bits/sec  [ 5] 11.00-12.00 sec 1017 KBytes 8.34 Mbits/sec  [ 5] 12.00-13.00 sec 1.61 MBytes 13.5 Mbits/sec  [ 5] 13.00-14.00 sec 1.61 MBytes 13.5 Mbits/sec  [ 5] 14.00-15.00 sec 1.71 MBytes 14.3 Mbits/sec  [ 5] 15.00-16.00 sec 1.57 MBytes 13.1 Mbits/sec  [ 5] 16.00-17.00 sec 1.76 MBytes 14.8 Mbits/sec  [ 5] 17.00-18.00 sec 1.65 MBytes 13.8 Mbits/sec  [ 5] 18.00-19.00 sec 1.44 MBytes 12.1 Mbits/sec  [ 5] 19.00-20.00 sec 1.68 MBytes 14.1 Mbits/sec  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Retr  [ 5] 0.00-20.00 sec 29.2 MBytes 12.2 Mbits/sec 0 sender  [ 5] 0.00-20.00 sec 29.1 MBytes 12.2 Mbits/sec receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

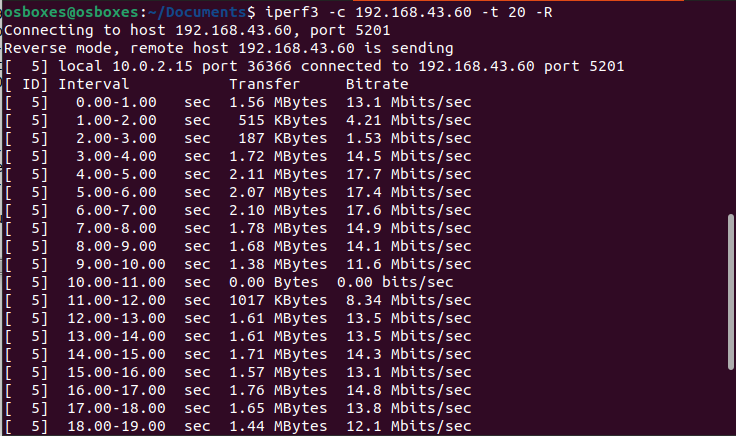


Figure : Talaria TWO Uplink TCP Throughput - Console Output

## Talaria TWO Downlink UDP Throughput

By default, iPerf3 performs throughput TCP test. However, it is possible to perform a UDP throughput test instead. This is accomplished by using UDP Traffic option -u when running iPerf3 Client using the T2 iPerf3 Server’s IP address:

|  |
| --- |
| iperf3 -c 192.168.43.60 –u -t 20 |

The test measures throughput from Client (PC) to Server (Talaria TWO) over TCP, which is ingress /downlink for the Talaria TWO.

The individual results are displayed in the Ubuntu PC iperf3 shell as well as in the Talaria TWO console. Results from Talaria TWO are shown in Ubuntu PC iPerf3 shell as well.

The default length of the buffer to read or write is 1460 and default bitrate is 1 Mbits/sec for UDP.

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 20  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 32967 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Total Datagrams  [ 5] 0.00-1.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 1.00-2.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 2.00-3.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 3.00-4.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 4.00-5.00 sec 127 KBytes 1.04 Mbits/sec 89  [ 5] 5.00-6.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 6.00-7.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 7.00-8.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 8.00-9.00 sec 127 KBytes 1.04 Mbits/sec 89  [ 5] 9.00-10.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 10.00-11.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 11.00-12.01 sec 128 KBytes 1.04 Mbits/sec 90  [ 5] 12.01-13.00 sec 127 KBytes 1.05 Mbits/sec 89  [ 5] 13.00-14.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 14.00-15.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 15.00-16.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 16.00-17.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 17.00-18.00 sec 127 KBytes 1.04 Mbits/sec 89  [ 5] 18.00-19.00 sec 128 KBytes 1.05 Mbits/sec 90  [ 5] 19.00-20.00 sec 128 KBytes 1.05 Mbits/sec 90  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-20.00 sec 2.50 MBytes 1.05 Mbits/sec 0.000 ms 0/1796 (0%) sender  [ 5] 0.00-20.00 sec 2.50 MBytes 1.05 Mbits/sec 0.000 ms 0/1796 (0%) receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

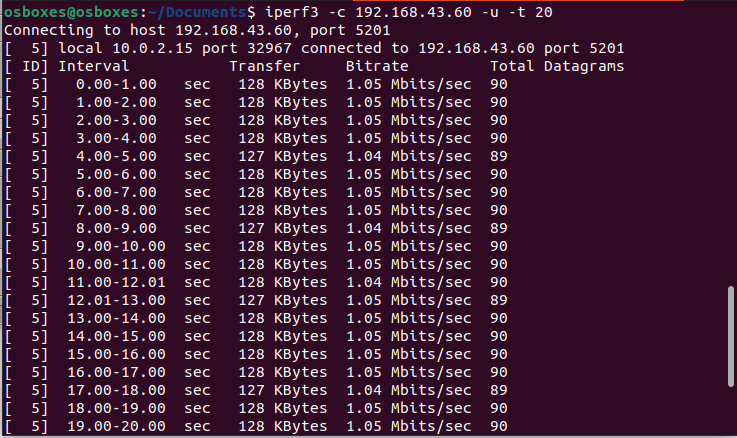


Figure : Talaria TWO Downlink UDP Throughput – Console Output

## Talaria TWO Uplink UDP Throughput

This is accomplished by using UDP Traffic option -u and Reverse Mode option -R together on the PC iPerf3 Client command line using the Talaria TWO iPerf3 Server’s IP address:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 20 -R |

As its run in Reverse Mode, the test measures throughput from Server (Talaria TWO) to Client (PC) over TCP, which is egress /uplink for the Talaria TWO.

The individual results are displayed in the Ubuntu PC iperf3 shell as well as in the Talaria TWO console. Results from Talaria TWO are shown in Ubuntu PC iPerf3 shell as well.

The default length of the buffer to read or write is 1460 and default bitrate is 1 Mbit/sec for UDP.

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 20 -R  Connecting to host 192.168.43.60, port 5201  Reverse mode, remote host 192.168.43.60 is sending  [ 5] local 10.0.2.15 port 47520 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-1.00 sec 133 KBytes 1.09 Mbits/sec 774.885 ms 1/94 (1.1%)  [ 5] 1.00-2.00 sec 127 KBytes 1.04 Mbits/sec 3.371 ms 0/89 (0%)  [ 5] 2.00-3.00 sec 128 KBytes 1.05 Mbits/sec 0.863 ms 0/90 (0%)  [ 5] 3.00-4.00 sec 127 KBytes 1.04 Mbits/sec 0.615 ms 0/89 (0%)  [ 5] 4.00-5.00 sec 128 KBytes 1.05 Mbits/sec 1.340 ms 0/90 (0%)  [ 5] 5.00-6.00 sec 128 KBytes 1.05 Mbits/sec 1.460 ms 0/90 (0%)  [ 5] 6.00-7.00 sec 127 KBytes 1.04 Mbits/sec 1.169 ms 0/89 (0%)  [ 5] 7.00-8.00 sec 128 KBytes 1.05 Mbits/sec 1.023 ms 0/90 (0%)  [ 5] 8.00-9.00 sec 128 KBytes 1.05 Mbits/sec 0.772 ms 0/90 (0%)  [ 5] 9.00-10.00 sec 127 KBytes 1.04 Mbits/sec 1.373 ms 0/89 (0%)  [ 5] 10.00-11.00 sec 128 KBytes 1.05 Mbits/sec 0.771 ms 0/90 (0%)  [ 5] 11.00-12.00 sec 125 KBytes 1.03 Mbits/sec 1.312 ms 1/89 (1.1%)  [ 5] 12.00-13.00 sec 128 KBytes 1.05 Mbits/sec 0.753 ms 0/90 (0%)  [ 5] 13.00-14.00 sec 128 KBytes 1.05 Mbits/sec 0.787 ms 0/90 (0%)  [ 5] 14.00-15.00 sec 127 KBytes 1.04 Mbits/sec 0.712 ms 0/89 (0%)  [ 5] 15.00-16.00 sec 127 KBytes 1.04 Mbits/sec 1.185 ms 1/90 (1.1%)  [ 5] 16.00-17.00 sec 128 KBytes 1.05 Mbits/sec 0.860 ms 0/90 (0%)  [ 5] 17.00-18.00 sec 127 KBytes 1.04 Mbits/sec 0.719 ms 0/89 (0%)  [ 5] 18.00-19.00 sec 128 KBytes 1.05 Mbits/sec 0.745 ms 0/90 (0%)  [ 5] 19.00-20.00 sec 125 KBytes 1.03 Mbits/sec 1.101 ms 1/89 (1.1%)  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-20.00 sec 2.50 MBytes 1.05 Mbits/sec 0.000 ms 0/1797 (0%) sender  [ 5] 0.00-20.00 sec 2.50 MBytes 1.05 Mbits/sec 1.101 ms 4/1796 (0.22%) receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

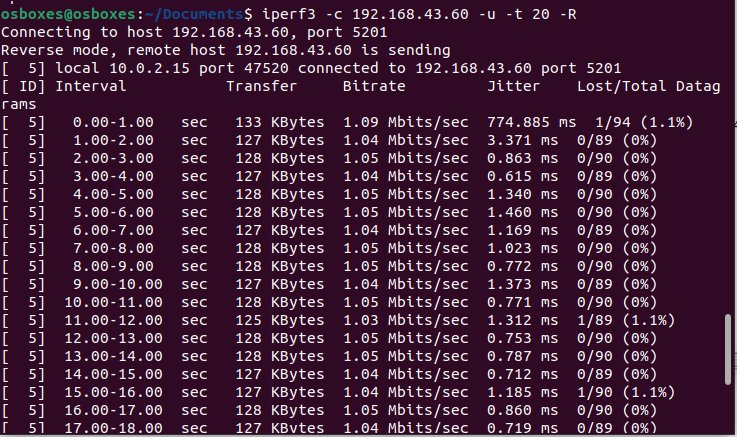


Figure : Talaria TWO Uplink UDP Throughput - Console Output

## Number of Streams

Option -P specifies the number of parallel Client streams to run. Default value is 1 if this option is not specified. All the tests specified in previous sections can be run with multiple client streams as well.

### Example UDP Downlink Throughput Test with 3 Parallel Streams

For example, UDP Downlink Throughput test with three parallel streams can be run with below command using -P3

|  |
| --- |
| iperf3 -c 192.168.43.60 –u -t 20 –P 3 |

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 –u -t 20 –P 3  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 36384 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Retr Cwnd  [ 5] 0.00-1.00 sec 1.39 MBytes 11.7 Mbits/sec 0 103 KBytes  [ 5] 1.00-2.00 sec 1.32 MBytes 11.0 Mbits/sec 0 101 KBytes  [ 5] 2.00-3.00 sec 1.81 MBytes 15.2 Mbits/sec 0 101 KBytes  [ 5] 3.00-4.00 sec 1.44 MBytes 12.1 Mbits/sec 0 101 KBytes  [ 5] 4.00-5.00 sec 2.02 MBytes 17.0 Mbits/sec 0 101 KBytes  [ 5] 5.00-6.00 sec 1.56 MBytes 13.1 Mbits/sec 0 101 KBytes  [ 5] 6.00-7.00 sec 1.78 MBytes 14.9 Mbits/sec 0 101 KBytes  [ 5] 7.00-8.00 sec 1.93 MBytes 16.2 Mbits/sec 0 101 KBytes  [ 5] 8.00-9.00 sec 1.81 MBytes 15.2 Mbits/sec 0 101 KBytes  [ 5] 9.00-10.00 sec 1.87 MBytes 15.7 Mbits/sec 0 101 KBytes  [ 5] 10.00-11.00 sec 1.59 MBytes 13.4 Mbits/sec 0 101 KBytes  [ 5] 11.00-12.00 sec 2.05 MBytes 17.2 Mbits/sec 0 101 KBytes  [ 5] 12.00-13.00 sec 1.16 MBytes 9.76 Mbits/sec 0 101 KBytes  [ 5] 13.00-14.00 sec 1.59 MBytes 13.4 Mbits/sec 0 101 KBytes  [ 5] 14.00-15.00 sec 1.78 MBytes 14.9 Mbits/sec 0 101 KBytes  [ 5] 15.00-16.00 sec 1.35 MBytes 11.3 Mbits/sec 0 101 KBytes  [ 5] 16.00-17.00 sec 1.65 MBytes 13.9 Mbits/sec 0 101 KBytes  [ 5] 17.00-18.00 sec 1.56 MBytes 13.1 Mbits/sec 0 101 KBytes  [ 5] 18.00-19.00 sec 1.84 MBytes 15.4 Mbits/sec 0 101 KBytes  [ 5] 19.00-20.00 sec 1.65 MBytes 13.9 Mbits/sec 0 101 KBytes  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Retr  [ 5] 0.00-20.00 sec 33.2 MBytes 13.9 Mbits/sec 0 sender  [ 5] 0.00-20.00 sec 32.6 MBytes 13.7 Mbits/sec receiver  iPerf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

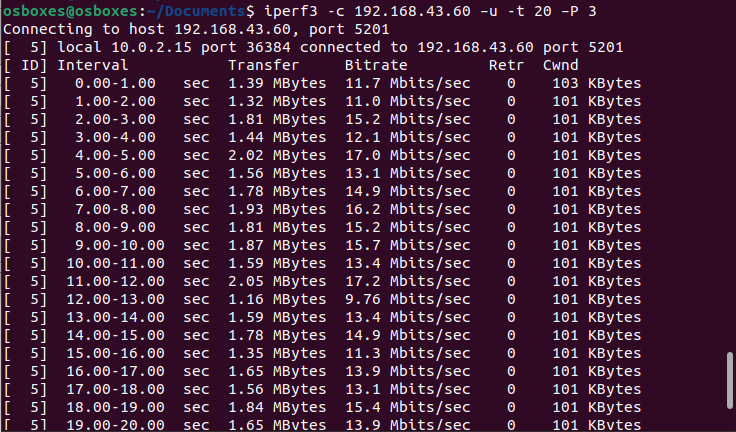


Figure : UDP Downlink Throughput Test with 3 Parallel Streams - Console Output

### Example TCP Uplink Throughput Test with 2 Parallel Streams

Another example here shows TCP Uplink Throughput test with two parallel streams, which can be run with below command using –P2

|  |
| --- |
| iperf3 -c 192.168.43.60 –u -t 5 –P 2 -R |

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 –u -t 5 –P 2 -R  Connecting to host 192.168.43.60, port 5201  Reverse mode, remote host 192.168.43.60 is sending  [ 5] local 10.0.2.15 port 36392 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate  [ 5] 0.00-1.00 sec 1.15 MBytes 9.61 Mbits/sec  [ 5] 1.00-2.00 sec 2.18 MBytes 18.3 Mbits/sec  [ 5] 2.00-3.00 sec 2.31 MBytes 19.4 Mbits/sec  [ 5] 3.00-4.00 sec 2.41 MBytes 20.2 Mbits/sec  [ 5] 4.00-5.00 sec 2.36 MBytes 19.8 Mbits/sec  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Retr  [ 5] 0.00-5.00 sec 10.5 MBytes 17.6 Mbits/sec 0 sender  [ 5] 0.00-5.00 sec 10.4 MBytes 17.5 Mbits/sec receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

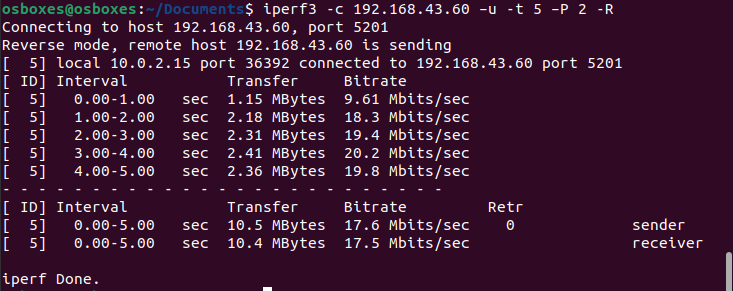


Figure : TCP Uplink Throughput Test with 2 Parallel Streams - Console Output

## Specific Transfer Size Test

Instead of specific duration with -t, it is possible to specify a certain amount of data to be transferred during an iPerf throughput test.

This is accomplished by using option -n to specify the number of bytes to transfer during the test, when running iPerf3 Client using the Talaria TWO iPerf3 Server’s IP address.

-n option takes number of bytes as input which can be specified in K or M.

For example, TCP Uplink Throughput test with total 10M Bytes data to be transferred can be run with below command using -n

|  |
| --- |
| iperf3 -c 192.168.43.60 –n 10M |

Console output from Ubuntu iPerf3 Client shell stops after sending specified bytes is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 –n 10M  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 36406 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Retr Cwnd  [ 5] 0.00-1.00 sec 944 KBytes 7.73 Mbits/sec 0 106 KBytes  [ 5] 1.00-2.00 sec 753 KBytes 6.17 Mbits/sec 0 106 KBytes  [ 5] 2.00-3.00 sec 1.81 MBytes 15.2 Mbits/sec 0 106 KBytes  [ 5] 3.00-4.00 sec 1.47 MBytes 12.3 Mbits/sec 0 106 KBytes  [ 5] 4.00-5.00 sec 1.72 MBytes 14.4 Mbits/sec 0 107 KBytes  [ 5] 5.00-6.00 sec 1.72 MBytes 14.4 Mbits/sec 0 107 KBytes  [ 5] 6.00-7.00 sec 1.01 MBytes 8.48 Mbits/sec 0 106 KBytes  [ 5] 7.00-8.00 sec 1.75 MBytes 14.6 Mbits/sec 0 106 KBytes  [ 5] 8.00-9.00 sec 1.59 MBytes 13.4 Mbits/sec 0 106 KBytes  [ 5] 9.00-10.00 sec 1.78 MBytes 14.9 Mbits/sec 0 106 KBytes  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Retr  [ 5] 0.00-10.00 sec 14.5 MBytes 12.2 Mbits/sec 0 sender  [ 5] 0.00-10.00 sec 13.9 MBytes 11.6 Mbits/sec receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

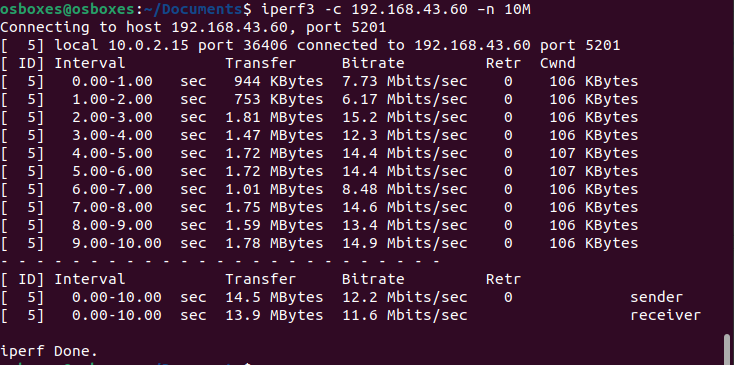


Figure : TCP Uplink Throughput test with 10M Bytes data – Console Output

## Specific Data Rate Test

It is possible to specify a datarate at which to perform the transfer during the throughput test.

This is accomplished by using option -b to specify the desired datarate for the test, when running iPerf3 Client using the Talaria TWO iPerf3 Server’s IP address.

-b option takes target bitrate in bits per second as input which can be specified in K or M. Default is 1 Mbit/sec for UDP, unlimited for TCP.

### Example TCP Downlink Throughput Test with Datarate of 2mbps

For example, TCP Downlink Throughput test with datarate of 2mbps can be run with below command using –b

|  |
| --- |
| iperf3 -c 192.168.43.60 -t 10 -b 2M |

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -t 10 -b 2M  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 36454 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Retr Cwnd  [ 5] 0.00-1.00 sec 359 KBytes 2.94 Mbits/sec 0 44.2 KBytes  [ 5] 1.00-2.00 sec 256 KBytes 2.10 Mbits/sec 0 37.1 KBytes  [ 5] 2.00-3.00 sec 128 KBytes 1.05 Mbits/sec 0 38.5 KBytes  [ 5] 3.00-4.00 sec 256 KBytes 2.10 Mbits/sec 0 42.8 KBytes  [ 5] 4.00-5.00 sec 256 KBytes 2.10 Mbits/sec 0 45.6 KBytes  [ 5] 5.00-6.00 sec 256 KBytes 2.10 Mbits/sec 0 54.2 KBytes  [ 5] 6.00-7.00 sec 256 KBytes 2.10 Mbits/sec 0 51.3 KBytes  [ 5] 7.00-8.00 sec 256 KBytes 2.10 Mbits/sec 0 57.0 KBytes  [ 5] 8.00-9.00 sec 256 KBytes 2.10 Mbits/sec 0 62.7 KBytes  [ 5] 9.00-10.00 sec 256 KBytes 2.10 Mbits/sec 0 61.3 KBytes  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Retr  [ 5] 0.00-10.00 sec 2.48 MBytes 2.08 Mbits/sec 0 sender  [ 5] 0.00-10.00 sec 2.48 MBytes 2.08 Mbits/sec receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

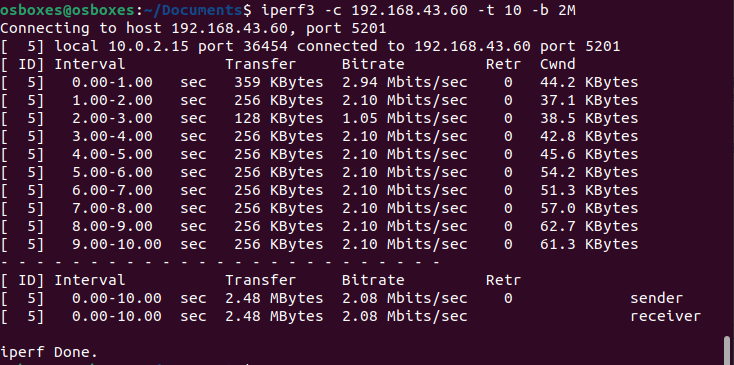


Figure : TCP Downlink Throughput Test with Datarate of 2mbps - Console Output

### Example UDP Uplink Throughput Test with Datarate of 2mbps

For example, UDP Uplink Throughput test with datarate of 2mbps can be run with below command using –b

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -b 2M -t 5 -R |

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -b 2M -t 5 -R  Connecting to host 192.168.43.60, port 5201  Reverse mode, remote host 192.168.43.60 is sending  [ 5] local 10.0.2.15 port 48755 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-1.00 sec 242 KBytes 1.99 Mbits/sec 12.535 ms 9/179 (5%)  [ 5] 1.00-2.00 sec 231 KBytes 1.89 Mbits/sec 1.357 ms 9/171 (5.3%)  [ 5] 2.00-3.00 sec 242 KBytes 1.99 Mbits/sec 0.826 ms 0/170 (0%)  [ 5] 3.00-4.00 sec 242 KBytes 1.98 Mbits/sec 1.058 ms 1/171 (0.58%)  [ 5] 4.00-5.00 sec 242 KBytes 1.99 Mbits/sec 0.461 ms 1/171 (0.58%)  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-5.00 sec 1.20 MBytes 2.02 Mbits/sec 0.000 ms 0/863 (0%) sender  [ 5] 0.00-5.00 sec 1.17 MBytes 1.97 Mbits/sec 0.461 ms 20/862 (2.3%) receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

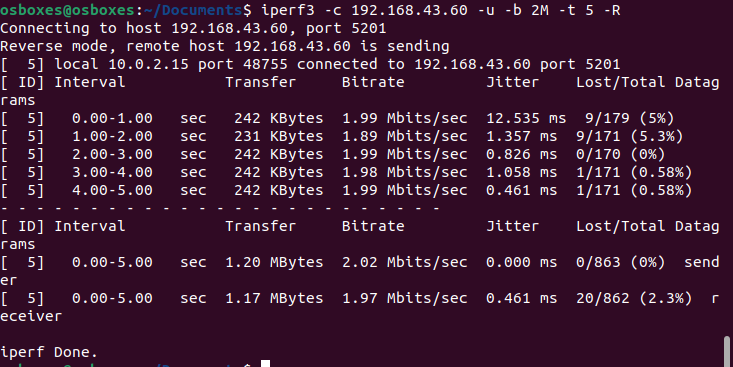


Figure : UDP Uplink Throughput Test with Datarate of 2mbps - Console Output

### Example UDP Downlink Throughput Test with Datarate of 3mbps

For example, UDP Downlink Throughput test with datarate of 3mbps can be run with below command using –b

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -b 3M -t 5 |

Console output from Ubuntu iPerf3 Client shell is as follows:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -b 3M -t 5  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 41918 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Total Datagrams  [ 5] 0.00-1.00 sec 366 KBytes 3.00 Mbits/sec 257  [ 5] 1.00-2.00 sec 366 KBytes 3.00 Mbits/sec 257  [ 5] 2.00-3.00 sec 366 KBytes 3.00 Mbits/sec 257  [ 5] 3.00-4.00 sec 366 KBytes 3.00 Mbits/sec 257  [ 5] 4.00-5.00 sec 366 KBytes 3.00 Mbits/sec 257  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-5.00 sec 1.79 MBytes 3.00 Mbits/sec 0.000 ms 0/1285 (0%) sender  [ 5] 0.00-5.00 sec 1.78 MBytes 2.98 Mbits/sec 0.000 ms 7/1285 (0.54%) receiver  iperf Done. |

Console output from Talaria TWO iPerf3 Server is as follows:

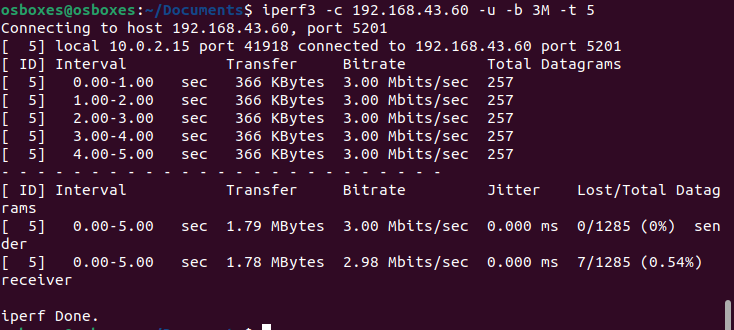


Figure : UDP Downlink Throughput Test with Datarate of 3mbps - Console Output

# Max Throughput Achieved over UDP

While running either TCP or UDP throughput test individually, we know that the TCP throughput is rate limited to a number not more than advertised window divided by network round trip time, as defined by the flow control mechanism associated with TCP.

Hence, it is common to compare max throughput achieved over UDP.

This can be measured by experimenting with different –b bitrate options and running UDP Downlink and Uplink tests. Below are some numbers measured in one such test setup. Please note that this can vary between different setups.

## UDP Downlink Test

Trying with different bitrates running downlink test, with bitrate set as 12 Mbits/sec, less than 1% data loss was achieved.

Command used on iPerf3 Client side is:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 10 -b 12M |

Output on Ubuntu console shows 11.8 Mbits/sec was achieved in this test setup with 2% data loss:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 10 -b 12M  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 55332 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Total Datagrams  [ 5] 0.00-1.00 sec 1.43 MBytes 12.0 Mbits/sec 1027  [ 5] 1.00-2.00 sec 1.43 MBytes 12.0 Mbits/sec 1027  [ 5] 2.00-3.00 sec 1.43 MBytes 12.0 Mbits/sec 1028  [ 5] 3.00-4.00 sec 1.43 MBytes 12.0 Mbits/sec 1027  [ 5] 4.00-5.00 sec 1.43 MBytes 12.0 Mbits/sec 1028  [ 5] 5.00-6.00 sec 1.43 MBytes 12.0 Mbits/sec 1027  [ 5] 6.00-7.00 sec 1.43 MBytes 12.0 Mbits/sec 1027  [ 5] 7.00-8.00 sec 1.43 MBytes 12.0 Mbits/sec 1028  [ 5] 8.00-9.00 sec 1.43 MBytes 12.0 Mbits/sec 1027  [ 5] 9.00-10.00 sec 1.43 MBytes 12.0 Mbits/sec 1028  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-10.00 sec 14.3 MBytes 12.0 Mbits/sec 0.000 ms 0/10274 (0%) sender  [ 5] 0.00-10.00 sec 14.0 MBytes 11.8 Mbits/sec 0.000 ms 209/10274 (2%) receiver  iperf Done. |

## UDP Uplink Test

Similarly, trying with different bitrates running uplink test with -R, bitrate of around 25 Mbits/sec was achieved with less than 1% data loss when bitrate was set to 30M.

Command used on iPerf3 Client side is:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 10 -b 30M –R |

Output on Ubuntu console shows 8.40 Mbits/sec was achieved in this test setup with 54% data loss:

|  |
| --- |
| iperf3 -c 192.168.43.60 -u -t 10 -b 30M –R  Connecting to host 192.168.43.60, port 5201  [ 5] local 10.0.2.15 port 37027 connected to 192.168.43.60 port 5201  [ ID] Interval Transfer Bitrate Total Datagrams  [ 5] 0.00-1.00 sec 3.57 MBytes 30.0 Mbits/sec 2567  [ 5] 1.00-2.00 sec 3.58 MBytes 30.0 Mbits/sec 2569  [ 5] 2.00-3.00 sec 3.58 MBytes 30.0 Mbits/sec 2568  [ 5] 3.00-4.00 sec 3.58 MBytes 30.0 Mbits/sec 2568  [ 5] 4.00-5.00 sec 3.58 MBytes 30.0 Mbits/sec 2569  [ 5] 5.00-6.00 sec 3.58 MBytes 30.0 Mbits/sec 2568  [ 5] 6.00-7.00 sec 3.58 MBytes 30.0 Mbits/sec 2569  [ 5] 7.00-8.00 sec 3.58 MBytes 30.0 Mbits/sec 2568  [ 5] 8.00-9.00 sec 3.58 MBytes 30.0 Mbits/sec 2569  [ 5] 9.00-10.00 sec 3.58 MBytes 30.0 Mbits/sec 2568  - - - - - - - - - - - - - - - - - - - - - - - - -  [ ID] Interval Transfer Bitrate Jitter Lost/Total Datagrams  [ 5] 0.00-10.00 sec 35.8 MBytes 30.0 Mbits/sec 0.000 ms 0/25683 (0%) sender  [ 5] 0.00-16.00 sec 16.0 MBytes 8.40 Mbits/sec 0.000 ms 13663/25174 (54%) receiver  iperf Done. |

# Factors Affecting Throughput

## TCP Window Size

The TCP receive window is the amount of data that a machine on the network is willing to buffer for a connection. The receiver notifies the sender of its window size, and the sender can only send that amount of data before it must wait for acknowledgement and a window update from the receiver. If the TCP receive window is configured to be too small, this can have a negative effect on the throughput of the connection. As TCP is bidirectional, each host has a receive window.

An appropriate window size can be determined by considering the bandwidth-delay product of the network between the hosts. The bandwidth-delay product is the product of the bandwidth (in bits per second) and round-trip delay (in seconds) of the network. Assuming the receiver acknowledges the data it receives right away, the bandwidth-delay product represents the amount of data the sender can transmit before it receives any acknowledgement due to delay in the network. The window size of the receiver should thus be at least the size of the bandwidth-delay product for optimal throughput. If the window size is less than this, the sender will reach the window size limit before receiving any acknowledgement, causing it to stop transmitting and thus underutilizing the link.

TCP window size is of concern in long fat networks (LFNs), i.e. networks with high throughput and/or high delay. LFNs have a large bandwidth-delay product and therefore can benefit from large TCP receive windows.

The window size can be configured with the boot argument lwip.tcp\_wnd.

It sets the TCP window size to the value of this boot argument multiplied by the TCP maximum segment size (1,460 bytes). Maximum value is 32, incremental in steps of 4. For example:

|  |
| --- |
| $python script/boot.py --device /dev/ttyUSB2 --speed 2456700 --reset=evk42 apps/iperf3/iperf3.elf ssid=<your\_ssid> passphrase=<your\_passphrase> hwaddr=<02:03:04:05:06:0a> iperf3.extra\_info=1 lwip.tcp\_wnd=32 |

Console output:

|  |
| --- |
| UART:NWWWWWWAEBuild $Id: git-f92bee540 $  ssid=ACT102571068294 passphrase=43083191 iperf3.extra\_info=1 lwip.tcp\_wnd=32  addr e0:69:3a:00:2c:3e  [10.379,311] CONNECT:00:5f:67:cd:c5:a6 Channel:6 rssi:-15 dBm  [11.223,365] MYIP 192.168.0.105  [11.223,414] IPv6 [fe80::e269:3aff:fe00:2c3e]-link  IPerf3 server @ 192.168.0.105  ----------------------------------------  Iperf3 TCP/UDP server listening on 5201  ----------------------------------------  Accepted connection from 192.168.0.101 port 1050  [58.745,191] iperf3: param = {"tcp":true,"omit":0,"time":20,"parallel":1,"len":131072,"pacing\_timer":1000,"client\_version":"3.7"}  [ 1] local 192.168.0.105 port 5201 connected to 192.168.0.101 port 1052  [78.820,085] iperf3: client\_result = {"cpu\_util\_total":0.32061508034327585,"cpu\_util\_user":0.025213145835174724,"cpu\_util\_system":0.29539699946291903,"sender\_has\_retransmits":1,"congestion\_used":"cubic","streams":[{"id":1,"bytes":32572600,"retransmits":0,"jitter":0,"errors":0,"packets":0,"start\_time":0,"end\_time":20.000591}]}  ------------------------------  [ ID] Interval Transfer Bitrate  [ 1] iperf3[S-RX-tcp]: 0.0-20 sec 30.7 MBytes 12.8 Mbits/sec  User: 9950311 (49%)  IRQ: 1192082 (5%)  Idle: 8910179 (44%)  [78.821,837] iperf3: server\_result = {"cpu\_util\_total": 54,"cpu\_util\_user": 49,"cpu\_util\_system": 5,"sender\_has\_retransmits": -1,"congestion\_used": "cubic","streams": [{"id":1,"bytes":32201760,"retransmits":0,"jitter":0,"errors":0,"packets":0,"start\_time":0,"end\_time":20.000629}]}  ----------------------------------------  Iperf3 TCP/UDP server listening on 5201  ---------------------------------------- |

## TCP Maximum Segment Size

The TCP Maximum Segment Size (MSS) is the maximum amount of data that a host supports in a single TCP segment. The MSS is limited by the Maximum Transmission Unit (MTU) of the link layer to which the host is connected, minus the size required for TCP and IP headers. MSS is often announced by each host when a TCP connection is established.

Generally, higher MSS values result in higher throughput because this results in a higher ratio of application data to overhead. However, increasing the MSS also results in increased memory requirements on the host.

The MSS on Talaria TWO has been configured in LwIP to be 1,460 bytes.

# Talaria TWOs TCP/IP Stack (LwIP)

Talaria TWO uses LwIP, an open-source, lightweight TCP/IP stack. LwIP has been compiled into the static library liblwip2.a available within the SDK for applications to link with. This section describes the memory options that were configured when LwIP was integrated into the Talaria TWO SDK. For additional information about LwIP on Talaria TWO, refer to the User Guide available with the SDK.

## Memory Options

The header file sdk/include/lwip/lwipopts.h contains the LwIP memory options that were used when building the LwIP library available in the SDK. Consider the following lines found in the file:

|  |
| --- |
| #define MEM\_LIBC\_MALLOC 1  #define MEMP\_MEM\_MALLOC 1 |

Since MEM\_LIBC\_MALLOC was 1 during LwIP compilation, LwIP has been built to use the C standard library functions malloc and free for dynamic memory allocation instead of the default LwIP custom allocator (which treats a statically allocated chunk of memory as its heap).

Furthermore, MEMP\_MEM\_MALLOC was also set to 1 for LwIP compilation. This means that memory for various structures used by LwIP will also be dynamically allocated from the heap instead of coming from fixed-sized memory pools residing in a statically allocated memory chunk.

LwIP has therefore been configured to use the C library heap for its memory allocation instead of performing this allocation from statically allocated memory chunks. For this reason, sufficient heap space must be maintained by an application for LwIp to function. The current amount of available heap space can be obtained by an application while it is running by calling the os\_avail\_heap() function.

For additional information on LwIP memory options, refer to the following LwIP Wiki Article:

<https://lwip.fandom.com/wiki/Lwipopts.h>