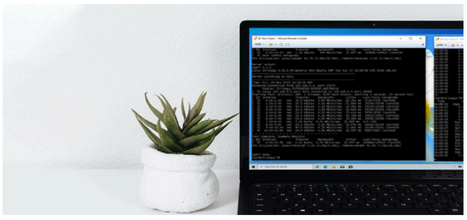
Knowing You're Network Ready



IPERF CAN BE WRONG!

HWe all love Open Source and I'm no exception.

Much of the world's computing is powered by Linux, including devices like your home router!

But this doesn't mean that it's perfect, and the chances are that the less people that use a particular bit of Open Source, the less well tested it is. So Linux gets a lot of use and has a lot of contributing developers, but tools like iPerf, widely used for network performance measurement, don't.

Why's he mentioning iPerf I hear you say, well it all started with an email from one of our customers…

Before I start though, a bit of background. At iTrinegy we make Network Emulation products. We also call them Software Defined Test Network products because that's so much more indicative of how our customers use them i.e. our customers want to create test networks, so that they can try out how their software, protocols or devices will work in particular networks environments. These test networks are "virtual" because you don't have to get access to an actual physical network, our Emulators recreate these conditions on demand.

So sometimes customers want to see how our software defined test network (i.e. our emulator) is behaving, especially when they're new to the product, and of course they need a measurement tool.

What to use? A quick google search, or asking a colleague, or prior knowledge nets the Open Source product iPerf – in fact iPerf3 in it's current guise.

So this customer – I'll call him John, emails in:

John:

"Hi Frank,

When we run a test without impairment [any network condition which makes the network not perfect…], the system works as expected. If we run a single impairment (e.g. bandwidth only, or latency only) we see consistent results.

But when we stack two impairments together we notice a significant drop in throughput, below the listed bandwidth.

For example, if we configure a 1Mbps bandwidth limit and also 5ms delay, we notice less than 1 Mbps throughput average. This may be expected for a significant latency delay (e.g. 800ms) but we assumed that a small consistent latency delay would not significantly hamper total/ average throughput but it apparently does.

Ideally, we would like to know if this is known and expected, or rather, if it is a limitation of our setup, or maybe it's the norm for all simulators."

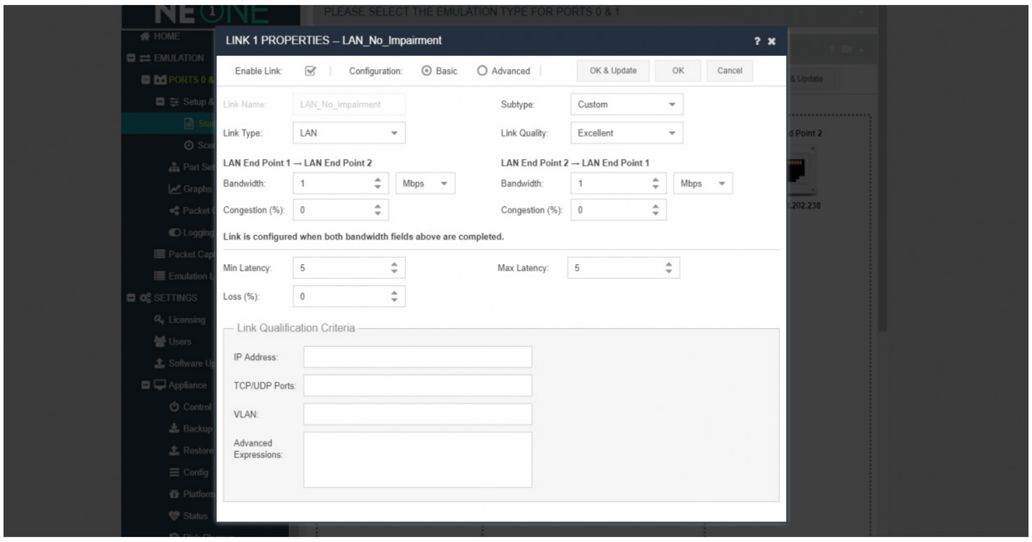
Me [thinking to myself….]:

"That's very odd. We routinely test things like this and have not seen this kind of issue. As most people run iPerf3 in TCP mode (the default) there can be some big misunderstandings about how latency affects throughput (i.e. Bandwidth used). But John is right, 5ms of latency is very little. So I decide to do the test myself."

Me:

"Hi John,

We can't find any issue with your parameter combination. Here are our test settings:


LINK 1 PROPERTIES – LAN_No_Impairment

Enable Link: ✓ | Configuration: [✓ Basic] Advanced

Link Name: LAN_No_Impairment
Link Type: LAN
Subtype: Custom
Link Quality: Excellent

LAN End Point 1 → LAN End Point 2
Bandwidth: 1 Mbps
Congestion (%): 0

LAN End Point 2 → LAN End Point 1
Bandwidth: 1 Mbps
Congestion (%): 0

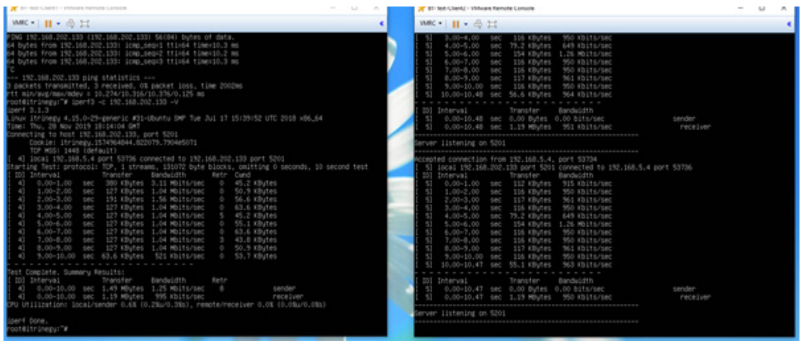
Link is configured when both bandwidth fields above are completed.

Min Latency: 5
Max Latency: 5
Loss (%): 0

Link Qualification Criteria
IP Address:
TCP/UDP Ports:
VLAN:
Advanced Expressions:

– which in fact makes the latency 10ms RTT – 5ms in each direction.

We used iPerf3 in TCP mode to do a test. Here are the screens: iPerf client on the left and iPerf server on the right. Client command: iperf3 -c 192.168.5.5, Server command: iperf3 -s



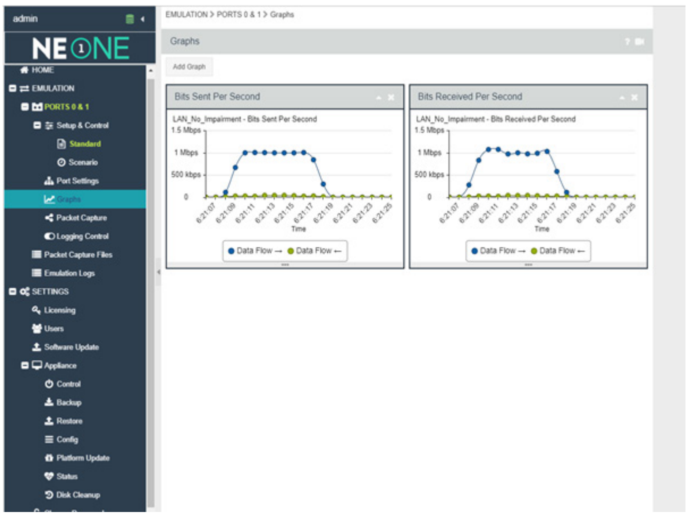
```
PING 192.168.202.133 (192.168.202.133) 56(84) bytes of data.
64 bytes from 192.168.202.133: icmp_seq=1 ttl=64 time=10.3 ms
64 bytes from 192.168.202.133: icmp_seq=2 ttl=64 time=10.2 ms
64 bytes from 192.168.202.133: icmp_seq=3 ttl=64 time=10.3 ms
^C
--- 192.168.202.133 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 10.274/10.316/10.376/0.125 ms
root@itrinegy:~# iperf3 -c 192.168.202.133 -V
iperf 3.1.3
Linux itrinegy 4.15.0-29-generic #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018 x86_64
Time: Thu, 28 Nov 2019 18:14:04 GMT
Connecting to host 192.168.202.133, port 5201
      Cookie: itrinegy.1574964844.822079.7904e5071
      TCP MSS: 1448 (default)
[  4] local 192.168.5.4 port 53736 connected to 192.168.202.133 port 5201
Starting Test: protocol: TCP, 1 streams, 131072 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   380 KBytes  3.11 Mbits/sec    0   45.2 KBytes
[  4]   1.00-2.00   sec   127 KBytes  1.04 Mbits/sec    0   50.9 KBytes
[  4]   2.00-3.00   sec   191 KBytes  1.56 Mbits/sec    0   56.6 KBytes
[  4]   3.00-4.00   sec   127 KBytes  1.04 Mbits/sec    0   63.6 KBytes
[  4]   4.00-5.00   sec   127 KBytes  1.04 Mbits/sec    5   45.2 KBytes
[  4]   5.00-6.00   sec   127 KBytes  1.04 Mbits/sec    0   55.1 KBytes
[  4]   6.00-7.00   sec   127 KBytes  1.04 Mbits/sec    0   63.6 KBytes
[  4]   7.00-8.00   sec   127 KBytes  1.04 Mbits/sec    3   43.8 KBytes
[  4]   8.00-9.00   sec   127 KBytes  1.04 Mbits/sec    0   50.9 KBytes
[  4]   9.00-10.00  sec  63.6 KBytes   521 Kbits/sec    0   53.7 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.49 MBytes  1.25 Mbits/sec    8             sender
[  4]   0.00-10.00  sec  1.19 MBytes   995 Kbits/sec                  receiver
CPU Utilization: local/sender 0.6% (0.2%u/0.3%s), remote/receiver 0.0% (0.0%u/0.0%s)

iperf Done.
root@itrinegy:~#
```

```
[  5]   3.00-4.00   sec   116 KBytes   950 Kbits/sec
[  5]   4.00-5.00   sec  79.2 KBytes   649 Kbits/sec
[  5]   5.00-6.00   sec   154 KBytes  1.26 Mbits/sec
[  5]   6.00-7.00   sec   116 KBytes   950 Kbits/sec
[  5]   7.00-8.00   sec   116 KBytes   950 Kbits/sec
[  5]   8.00-9.00   sec   117 KBytes   961 Kbits/sec
[  5]   9.00-10.00  sec   116 KBytes   950 Kbits/sec
[  5]  10.00-10.48  sec  56.6 KBytes   964 Kbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.48  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-10.48  sec  1.19 MBytes   951 Kbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.5.4, port 53734
[  5] local 192.168.202.133 port 5201 connected to 192.168.5.4 port 53736
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec   112 KBytes   915 Kbits/sec
[  5]   1.00-2.00   sec   116 KBytes   950 Kbits/sec
[  5]   2.00-3.00   sec   117 KBytes   961 Kbits/sec
[  5]   3.00-4.00   sec   116 KBytes   950 Kbits/sec
[  5]   4.00-5.00   sec  79.2 KBytes   649 Kbits/sec
[  5]   5.00-6.00   sec   154 KBytes  1.26 Mbits/sec
[  5]   6.00-7.00   sec   116 KBytes   950 Kbits/sec
[  5]   7.00-8.00   sec   116 KBytes   950 Kbits/sec
[  5]   8.00-9.00   sec   117 KBytes   961 Kbits/sec
[  5]   9.00-10.00  sec   116 KBytes   950 Kbits/sec
[  5]  10.00-10.47  sec  55.1 KBytes   963 Kbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.47  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-10.47  sec  1.19 MBytes   950 Kbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

– You can see that the bandwidth is regulated correctly (I'm not sure why iPerf3 shows a difference on the client and server side bandwidths).

and here are the graphs from the NE-ONE Professional's graphing menu option:



You can see that the client sends data faster than 1Mbps (right hand graph – Bits Received Per Second) – typical TCP. 'We' then regulate the bandwidth (and queue) and send it out to the server (left hand graph Bit's Sent per second) – notice that it is smoother. [The difference between the graphs is that the received graph is pre-impairment and the sent graph is post-impairment] – The green line is the reverse flow direction – ACKs.

So no problem here."

John:

"Ah, we actually run the [iperf3] command using UDP packets, for 10 seconds, set to 1Gbps (regardless of bandwidth impairment setting on emulator). We limit UDP datagrams to 1000 bytes to avoid processing delays on the receiving end due to fragmentation."

Me: [Thinking to myself…]

"Ah, UDP usually creates fewer misunderstandings than TCP (due to the lack of flow regulation which TCP has). I wonder what's going on?"

Me:

"Please can you send us your iPerf3 command lines"

John:

"Here is client-side command example:

iperf3 -V -u -b 1Gbps -f m -t 10 –length 1000 -c <IPADDRESS>

Server is standard but with verbose:

Iperf3 -s -f m -V

My NE-ONE Professional configuration is a copy of the "LAN_NO_IMPAIRMENT" sample, modified with changes to latency and bandwidth."

Me [Testing…]

I set up the test with John's commands and find that, yes, iPerf3 is reporting, at the server side, that we're only transmitting 0.33 (0.20, 0.26…. – see below) Mbps, not ~1Mbps as required. However, our live graphs correctly show ~1Mbps though. Very odd indeed…

I go to our developers, who say that they routinely test this, and given that our graphs are showing the correct values they suggest that iperf3 is reporting the wrong values. They ask me to find an independent method of verifying the bandwidth. I look at using tcpdump (on the server) and analysing the pcap file produced, but in the end for simplicity decide on a tool that will report on Ethernet interface I/O statistics. I try 3 of them and all agree that the NE-ONE Professional is working correctly, but in the end I settle on ifstat as it produces an easy to read continuous output. Time to report my findings to John.

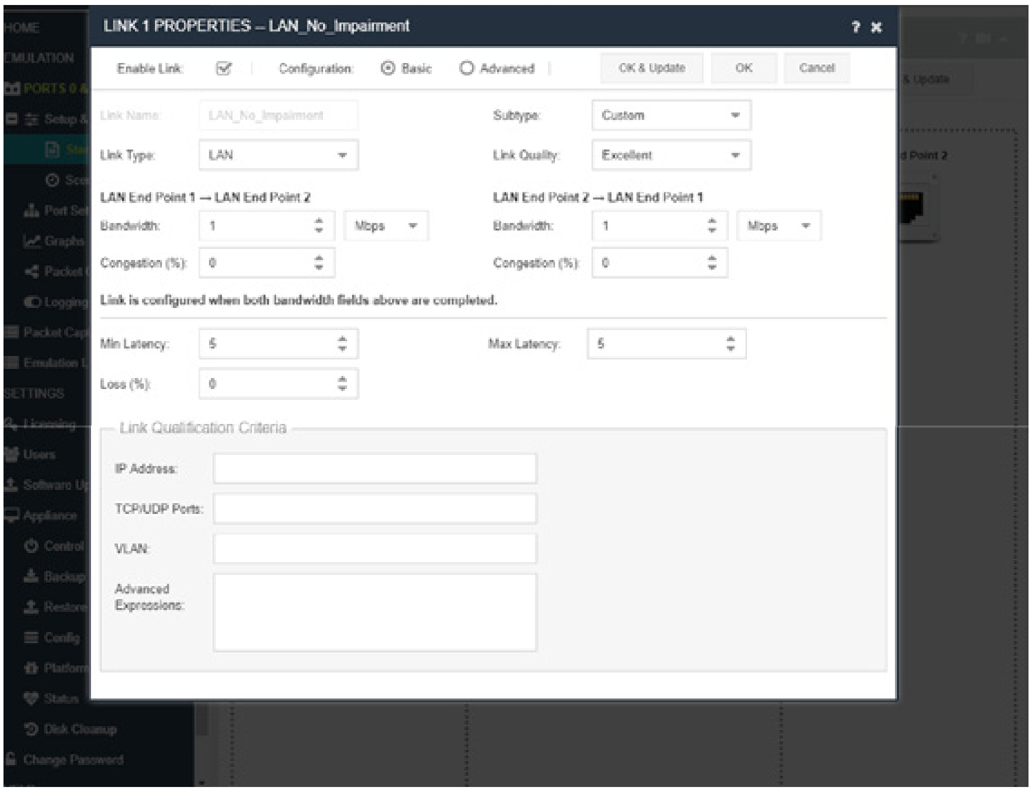
Me:

"Hi John,

Thanks very much for your command line.

I ran your test and got some weird results – just as you said. I went to our developers for advice and they strongly suggested that iPerf3 was wrong and asked me to monitor the packets at the server end. After a bit of trial and error I settled on using the tool ifstat (sudo apt install ifstat). And yes iPerf3 is getting it wrong. Here's my setup:

NE-ONE Professional:



LINK 1 PROPERTIES – LAN_No_Impairment

Enable Link: ✓ Configuration: [✓ Basic] Advanced

OK & Update | OK | Cancel

Link Name:	LAN_No_Impairment	Subtype:	Custom
Link Type:	LAN	Link Quality:	Excellent

LAN End Point 1 → LAN End Point 2		LAN End Point 2 → LAN End Point 1	
Bandwidth:	1 Mbps	Bandwidth:	1 Mbps
Congestion (%):	0	Congestion (%):	0

Link is configured when both bandwidth fields above are completed.

Min Latency:	5	Max Latency:	5
Loss (%):	0		

Link Qualification Criteria

IP Address:

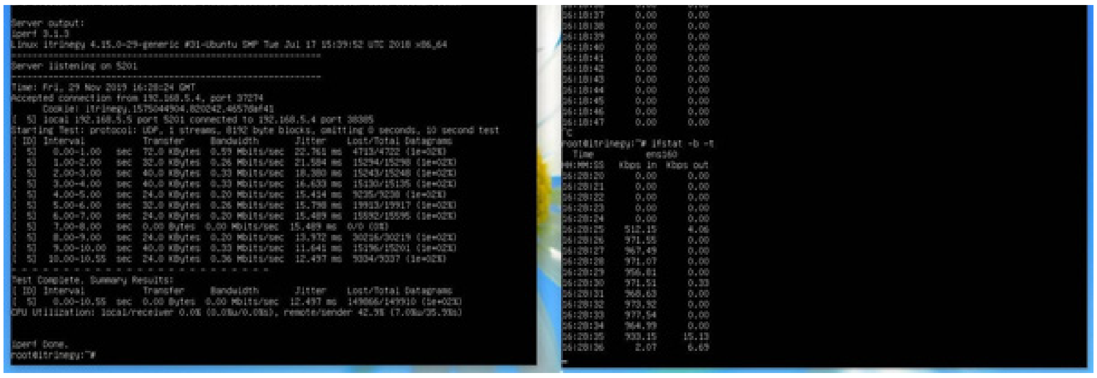
TCP/UDP Ports:

VLAN:

Advanced Expressions:

Here's the client command I used (as you specified plus -get-server-output so that we could see the server end output): iperf3 -c 192.168.5.5 -u -b 1Gbps -f m -t 10 -length 1000 -get-server-output I "backgrounded" the iperf server: iperf3 -s -V -f m >iperf3_s.out & (so that I could use ifstat -b -t in the foreground).

Here are the results (client on left, server on right:



```
Server output:
iperf 3.1.3
Linux itrinegy 4.15.0-29-generic #31-Ubuntu SMP Tue Jul 17 15:39:52 UTC 2018 x86_64
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Time: Fri, 29 Nov 2019 16:28:24 GMT
Accepted connection from 192.168.5.4, port 37274
      Cookie: itrinegy.1575044904.820242.46578af41
[  5] local 192.168.5.5 port 5201 connected to 192.168.5.4 port 38385
Starting Test: protocol: UDP, 1 streams, 8192 byte blocks, omitting 0 seconds, 10 second test
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec  72.0 KBytes  0.59 Mbits/sec  22.761 ms  4713/4722 (1e+02%)
[  5]   1.00-2.00   sec  32.0 KBytes  0.26 Mbits/sec  21.584 ms  15294/15298 (1e+02%)
[  5]   2.00-3.00   sec  40.0 KBytes  0.33 Mbits/sec  18.380 ms  15243/15248 (1e+02%)
[  5]   3.00-4.00   sec  40.0 KBytes  0.33 Mbits/sec  16.633 ms  15130/15135 (1e+02%)
[  5]   4.00-5.00   sec  24.0 KBytes  0.20 Mbits/sec  15.414 ms  9235/9238 (1e+02%)
[  5]   5.00-6.00   sec  32.0 KBytes  0.26 Mbits/sec  15.798 ms  19913/19917 (1e+02%)
[  5]   6.00-7.00   sec  24.0 KBytes  0.20 Mbits/sec  15.489 ms  15592/15595 (1e+02%)
[  5]   7.00-8.00   sec  0.00 Bytes  0.00 Mbits/sec  15.489 ms  0/0 (0%)
[  5]   8.00-9.00   sec  24.0 KBytes  0.20 Mbits/sec  13.972 ms  30216/30219 (1e+02%)
[  5]   9.00-10.00  sec  40.0 KBytes  0.33 Mbits/sec  11.641 ms  15196/15201 (1e+02%)
[  5]  10.00-10.55  sec  24.0 KBytes  0.36 Mbits/sec  12.497 ms  9334/9337 (1e+02%)
- - - - - - - - - - - - - - - - - - - - - - - - -
Test Complete. Summary Results:
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-10.55  sec  0.00 Bytes  0.00 Mbits/sec  12.497 ms  149866/149910 (1e+02%)
CPU Utilization: local/receiver 0.0% (0.0%u/0.0%s), remote/sender 42.3% (7.0%u/35.3%s)


iperf Done.
root@itrinegy:~#
```

```
16:18:37       0.00      0.00
16:18:38       0.00      0.00
16:18:39       0.00      0.00
16:18:40       0.00      0.00
16:18:41       0.00      0.00
16:18:42       0.00      0.00
16:18:43       0.00      0.00
16:18:44       0.00      0.00
16:18:45       0.00      0.00
16:18:46       0.00      0.00
16:18:47       0.00      0.00
^C
root@itrinegy:~# ifstat -b -t
  Time           ens160
HH:MM:SS    Kbps in  Kbps out
16:28:20       0.00      0.00
16:28:21       0.00      0.00
16:28:22       0.00      0.00
16:28:23       0.00      0.00
16:28:24       0.00      0.00
16:28:25     512.15      4.06
16:28:26     971.55      0.00
16:28:27     967.49      0.00
16:28:28     971.07      0.00
16:28:29     956.81      0.00
16:28:30     971.51      0.33
16:28:31     968.63      0.00
16:28:32     973.92      0.00
16:28:33     977.54      0.00
16:28:34     964.99      0.00
16:28:35     933.15     15.13
16:28:36       2.07      6.69
```

You can see that iPerf3 is saying that it's only receiving 0.26Mbps, for example in second 5-6, but clearly Linux (using ifstat) says it's getting 956.81 Kbps in that interval. This is very similar to the other intervals, only the first and last show appreciably less data (as you would expect).

So I believe that the NE-ONE Professional is working correctly here and that iPerf3 isn't.

If you'd like to verify this for yourself then load up ifstat or something similar and try it out.

————

John later confirms my results, and what we're both surprised at is why iPerf3 shows different results with 5ms each way latency compared to 0ms (in UDP mode at a 1Mbps throughput limit) – it must be something to do with the way it's calculating the bandwidth received at the server side. In this case it is wrong though – our graphs plus ifstat (and Nload, bmon, slurm etc) all agree on that.

Of course, John's purpose in using our NE-ONE Professional is not to test iPerf3 or vice-versa, so having ascertained that the NE-ONE Professional is verifiably working as it should he moves onto the real applications under test…

So, I think the moral of this story is that while open source tools can be useful, you have to factor in that, unless there is a very active community supporting and correcting them (as with Linux) you're on your own.

iTRINEGY Knowing You're Network Ready

Ref: CStudy-Helyx-21042021