09-02-2023

White Paper:
Network Performance Tests over the 100G BELLA Link between GÉANT and RNP

Grant Agreement No.:	856726
Work Package	WP6
Task Item:	Task 3
Document Type:	White Paper
Dissemination Level:	PU (Public)
Lead Partner:	Jisc
Document ID:	GN4-3-22-T5D0T7
Authors:	Raul Lopes (Jisc), Duncan Rand (Jisc), Tim Chown (Jisc), Ivana Golub (PSNC), Pavle Vuletić (UoB/AMRES)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

Abstract

This white paper reports on network performance tests carried out in the GÉANT GN4-3 project in late 2021 and early 2022 over the newly established 100G BELLA research and education network link between Europe and South America. The tests, conducted to minimise impact on production traffic, demonstrated throughput of up to 96.9 Gbit/s using *iperf2* with optimised TCP parameters.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This white paper presents the results of a series of network performance tests run by the Monitoring and Management Task (Task 3) of the Network Technologies and Services Development Work Package (WP6) of the GÉANT GN4-3 project over the 100G BELLA (Building the Europe Link to Latin America) link between the European GÉANT and Brazilian RNP research and education networks in late 2021 and early 2022.

The document presents considerations for achieving high throughput, particularly for higher round-trip time (RTT) links. In our case, the tests show that throughput of over 96 Gbit/s can be achieved between Europe and Brazil on the new production BELLA link.

Such results require appropriate tuning of the data transfer nodes (DTNs) involved in the tests, including the Transmission Control Protocol (TCP) window size (dependent on the RTT), the number of parallel TCP streams used by the application, the TCP congestion algorithm, and the end-to-end path Maximum Transmission Unit (MTU) size. While we present here the tuning options that worked well for us, the authors encourage other users to experiment with tuning parameters to find optimal settings for their own specific scenarios.

While the document's focus is on testing network performance over the BELLA link, it also shows the value of other experimental capabilities. One is the ability for the perfSONAR network performance test platform to remotely initiate tests between two separate servers, acting as a third-party orchestrator; this removes the need for direct access for testing, but does require knowledge on how various parameters can be tuned. The other is the value of streaming telemetry for studying the fine-grained detail of traffic throughput over time; the detail shown in this white paper would not typically be seen in a traditional SNMP-based monitoring solution where five-minute traffic averages are the norm.

1 Introduction

In this white paper we present the results of work undertaken by the Monitoring and Management Task (Task 3) of the Network Technologies and Services Development Work Package (WP6) of the GÉANT GN4-3 project in late 2021 and early 2022 on testing network performance over the 100G BELLA (Building the Europe Link to Latin America) circuit between GÉANT and Fortaleza in RNP, the Brazilian National Research and Education Network (NREN).

The driver for publishing these results is that an increasing number of institutions in the GÉANT community are connecting to their NREN networks at 100 Gbit/s and are seeking to optimise the performance of large-scale data transfers over those links. While network testing using tools such as perfSONAR [perfSONAR] and *iperf2* [iperf2] or *iperf3* [iperf3] is commonplace at 10 Gbit/s, testing at higher line rates is not such a well-trodden path. In particular, network testing at speeds equal to or less than 10 Gbit/s doesn't require much tuning of the tools or server parameters and thus full-capacity operation can usually be readily achieved, whereas on 100 Gbit/s links more careful tuning is required to achieve full-capacity operation.

The link over which the tests were to be run was already carrying production traffic at the time of testing. It was therefore important to run the tests in a way that was considerate of this, by keeping the duration of each test short, typically 30–60 seconds, enough to be indicative of potential without filling the link for extended periods. The tools used to send test traffic were *iperf2* and *iperf3*, using TCP such that the test streams would back off in the event of congestion.

The utilisation of the BELLA link as observed through the GÉANT BRIAN monitoring platform [BRIAN_BELLA] was generally under 20 Gbit/s in the period leading up to the testing. Thus we would expect a good test to reach 80 Gbit/s, and, if the link were quiet, that we could reach higher throughput. It is important to note that the throughput achieved will vary with the background traffic present at the time.

The scope of the reported tests included tuning the hosts used for the tests and exploring the effect of MTU size, TCP congestion algorithms, window sizes, and the number of parallel streams used on the rates achieved.

The test endpoints were systems running perfSONAR provided by SURF and RNP. The open-source perfSONAR toolkit is widely used in R&E networks, most notably by the CERN experiment community for monitoring the network infrastructure used by the Worldwide Large Hadron Collider Computing Grid (WLCG).

One very useful feature of perfSONAR is that, subject to the appropriate permissions being set, it can be used, via its built-in *pscheduler* command, to remotely run *iperf2*, *iperf3* and *traceroute* (or other)

tests between remote endpoints. We were thus able to run tests from SURF to RNP from a perfSONAR instance on the Janet network. It is also possible to request various tuning options to be set for those remote tests, as discussed later in this document.

While the focus of this report is the 100G BELLA testing, it also serves to highlight the potential of streaming telemetry to view fine-grained detail of network utilisation. A prototype (non-production) streaming telemetry system was kindly provided by SURF, which was used to validate the observed throughput and to collect the plots that are included in this document.

The use of streaming telemetry is potentially very useful when considering short-lived throughput tests such as those run in these experiments; utilisation over time can be seen for very short periods, for example over a 30-second window, where a typical SNMP-based approach will only present 5-minute averages of utilisation.

The early results from these tests were presented at the 2nd Performance Management Workshop, organised by WP6 of the GÉANT GN4-3 project, in March 2022 [PMW_Mar2022]. SURF presented on streaming telemetry at the 2nd Telemetry and Data Workshop in April 2022, where Jisc also compared the uses of SNMP and streaming telemetry-based monitoring [T&DW_Apr2022].

We would like to acknowledge the assistance of Marcos Schwartz and Jeferson Souza of RNP, Max Mudde at SURF, and David Richardson at Jisc in making these tests possible.

This document is organised as follows: Section 2 introduces the BELLA link and shows the forward path between the SURF and RNP servers. Section 3 describes the configurations used for the tests, including the servers, *pscheduler*, test tools, TCP congestion algorithms and window sizes, and MTU size. Section 4 presents the results of the tests, including the initial BELLA experiments and how we achieved maximum throughput, and also outlines future work. Section 5 recaps the key points and conclusions.

2 BELLA Link

The BELLA (Building the Europe Link to Latin America) link [BELLA] provides direct connectivity between Europe (GÉANT) and South America (RNP in Brazil). The link was upgraded from 10 Gbit/s to 100 Gbit/s in 2021, creating the potential for significant improvements to the capacity for research and education collaboration between the two regions.

The link was officially launched in September 2021 [BELLA_Launch], while further expansion of the connectivity within South America was reported in *CONNECT* magazine in January 2022 [BELLA_Expansion]. Figure 2.1 shows the BELLA connectivity from Portugal to Brazil in context, from the RedClara web site [BELLA_Map].

Figure 2.1: The BELLA link between Portugal and Brazil

The tests over the 100G BELLA link were run from a server in the SURF network to a server in the RNP network in Fortaleza. As mentioned above, the tests run were of short duration, to minimise the impact on production traffic.

During the tests the forward path between the SURF server and the server in RNP was confirmed by running a *pscheduler* trace task between the two perfSONAR systems:

```
$ pscheduler task trace --dest nfor.rnp.br --source ps2.netherlight.net

1 et0-0-0.1421.jnr01.asd001a.surf.net (145.146.0.5) AS1103 0.7 ms
     SURFNET-NL SURFnet, The Netherlands, NL
2 surfnet-gw.mx1.ams.nl.geant.net (62.40.124.39) AS20965 2 ms
     GEANT The GEANT IP Service, NL
3 surfnet.mx1.ams.nl.geant.net (62.40.124.38) AS20965 0.6 ms
     GEANT The GEANT IP Service, NL
4 ae7.mx1.fra.de.geant.net (62.40.98.187) AS20965 7.6 ms
     GEANT The GEANT IP Service, NL
5 ae3.mx1.gen.ch.geant.net (62.40.98.181) AS20965 15.6 ms
     GEANT The GEANT IP Service, NL
6 ae4.rt1.mar.fr.geant.net (62.40.98.237) AS20965 22.7 ms
     GEANT The GEANT IP Service, NL
7 ae5.mx1.mad.es.geant.net (62.40.98.226) AS20965 44.9 ms
     GEANT The GEANT IP Service, NL
8 ae4.mx2.lis.pt.geant.net (62.40.98.96) AS20965 45.6 ms
     GEANT The GEANT IP Service, NL
9 redclara-gw.lis.pt.geant.net (62.40.127.151) AS20965 108 ms
     GEANT The GEANT IP Service, NL
10 200.0.206.71 AS27750 107.3 ms
     Cooperacion Latino Americana de Redes Avanzadas, UY
```

The round-trip time (RTT) of the path under test was therefore approximately 100–110 ms. This property is important when considering the bandwidth-delay product (BDP) of the path, and the maximum volume of data that can be in flight at any one time. This has an impact on the server tuning required to optimise throughput, including the TCP window sizing, as described below.

3 Configurations for the Tests

In this section we describe the test systems and their configuration.

There is good guidance on tuning hosts and network elements for high-performance networking at the ESnet Fasterdata Knowledge Base [ESnet Fasterdata], which we have drawn upon. The Fasterdata guidance was initially published for 10G networks, but has since been updated.

One important principle the Fasterdata site mentions is that rather than seeking to push data as fast as possible over a single TCP stream, it is generally more effective and practical to use multiple TCP streams, each running at a lower rate. As discussed later in the document, there are practical limits to TCP window sizes, and thus to single stream rates that can be achieved, especially for high RTT paths. Another advantage of the multi-stream approach is that any packet loss on one stream does not impact the rate at which the other streams continue to send; only the impacted stream backs off when it observes loss (and thus likely congestion).

Data transfer applications such as GridFTP [GridFTP] typically use multiple TCP streams for this reason, while other applications may use multiple single-stream file transfers in parallel. In practice, user applications will have varying traffic and stream profiles, and traffic may even flow over many relatively small flows between multiple nodes in clusters at each end. The tests reported here are not intended to reflect the behaviour of any specific application, rather to show what is possible.

3.1 Test Servers

The test endpoints were servers running perfSONAR provided by SURF (*ps2.netherlight.net*) and RNP (*nfor.rnp.br*). The host at SURF was a Linux CentOS7 server with one CPU with four cores, eight hyper-threads, and 32 GB of RAM. All tests in this report were run from SURF towards RNP.

The SURF server was connected through to the European BELLA endpoint in Portugal via the GÉANT network (as indicated by the earlier traceroute and Figure 2.1), while the RNP server was within the RNP network near the Brazilian endpoint.

3.2 Using *pscheduler* to Execute Tests

The tests were initiated remotely by a third-party perfSONAR host on the Janet network by running *pscheduler*, the test scheduling tool provided as part of the perfSONAR toolkit.

The ability for perfSONAR, subject to configured controls, to run third-party tests in this way is a valuable property of the tool, particularly when debugging international, multi-domain network performance issues. Most perfSONAR servers in the R&E networks are not restricted, so such remote tests can be initiated, often to home in on where a specific problem might lie along a path.

Using *pscheduler* meant there was no need for the SURF and RNP staff to react to emails requesting local configuration changes on their servers. Rather, we sought to run all the tests, and to change the parameters on the tests, purely through *pscheduler*. Our experiences running the tests this way may be of interest to others looking to explore similar tuning options for specific network paths where perfSONAR is deployed.

The *pscheduler* options we used, which are discussed further in the following sections, were:

- `--congestion CWA` – to set the TCP congestion algorithm
- `-w WINDOW_SIZE` – to set the TCP window size
- `-m MSS` – to set the TCP Maximum Segment Size (MSS), which is closely related to the MTU
- `-b BANDWIDTH` – to rate limit a given stream
- `-P` – to specify the number of parallel TCP streams to use

While the MTU cannot be directly set, in practice setting the MSS should have a similar effect to setting the MTU, allowing for the IP header length.

Note that the `-w WINDOW_SIZE` option doesn't necessarily behave as you might expect. If window scaling is set to 1 for both servers at each end of the test, as was the case for our experiments, the actual window size used is double that requested, and *pscheduler* will warn about this in its output. It's therefore necessary to check the server settings and *pscheduler* output to be sure of the setting that is being used.

In the results presented in Section 3.5 and in Section 4, the reported window sizes are those actually used, and those were requested using a `-w` option of half that amount, so for example using `-w 128M` would result in a window of 256 MB.

There are other *pscheduler* options and examples listed within the perfSONAR documentation on throughput testing [perfSONAR_Throughput], and further tips on using *pscheduler* at the Fasterdata site [ESnet_Fasterdata_pscheduler].

3.3 Test Tools

The test tools that were available to be run from the SURF server via *pscheduler* included *iperf2* [iperf2] and *iperf3* [iperf3] for throughput tests and *traceroute* for routing diagnostics. There is also support in more recent versions of perfSONAR to use *ethr* for throughput tests, which is an option worth considering where it is installed.

The *iperf2* and *iperf3* tools are distinct and separately maintained. The *iperf2* tool supports multi-stream tests. In contrast, *iperf3* is single-threaded; this means that tests may be CPU-bound on some hosts, or on 40G/100G Network Interface Cards (NICs) [ESnet_Fasterdata_iperf]. It is possible with

iperf3 to use hyper-threading on two cores (running multiple logical cores on one core), whereby performance can be as much as doubled with two streams.

The *ethr* [ethr] throughput test is the one most recently added to perfSONAR. It can be performant but has limited output on RTT, losses, and retries, which means its utility is not as strong as it should be.

A point to note about transfer test tools, including *iperf2*, *iperf3* and *ethr*, is that they can sometimes be inconsistent when it comes to reporting bytes transferred and received. Counters for transferred and received bytes can be obtained from the operating system; in particular, *ethtool* is a tool recommended by the Linux *kernel.org* for, amongst its many uses, collecting reliable network interface data. In tests run in the Jisc test lab, we have observed some occasional discrepancies between what the test tools report and what *ethtool* shows. In one case we saw over 120 Gbit/s reported over a 100G path in the lab. It might be worth exploring these differences in future work. We saw no discrepancies in the traffic levels reported by the SURF streaming telemetry platform.

3.4 TCP Congestion Algorithms and Window Sizes

There are many variants of TCP congestion algorithms that ship as standard with Linux distributions. A server can be configured to run with any installed, supported variant, with its particular congestion-handling algorithms. In these tests we included Reno [Reno], CUBIC [CUBIC]and H-TCP [H-TCP]. A discussion of TCP algorithms can be found in [Dordal].

From time to time new variants of TCP emerge. TCP-BBRv2 [TCP_BBRv2] is a particularly promising new version, given it has been shown to be more performant under loss while also being 'friendly' to other non-BBRv2 algorithms running over the same links. ESnet gave a very good report on their early findings of BBRv2 at TNC22 [ESnet_BBRv2] using patches available from Google (BBRv2 is not included at the time of writing in mainstream Linux distributions). The ESnet report includes comparisons with Reno and CUBIC.

Another TCP configuration choice is the default size of buffers used, as set for the kernel. Modern Linux distributions can adjust buffer sizes dynamically, but setting the optimal size directly is generally good practice.

As mentioned earlier, to tune buffer sizing, the BDP of a link should be considered. The BDP, which reflects the maximum volume of data in flight, is expressed as a multiple of the link bandwidth and the RTT of the path.

The TCP throughput calculator provided by SWITCH is a useful resource for checking BDP and the resulting window sizes required [SWITCH_TCP_Throughput]. It also includes a throughput calculator based on MTU, RTT and loss using the Mathis et al. formula [Mathis]. As an example, for 100,000 Mbps (100 Gbit/s) over a 100 ms RTT, the calculator suggests a window size of 1,220 MB.

While the SWITCH tool can help give an idea of the ideal window size for a link, it has limitations as it ignores limits in the hardware and operating systems. The above example of 1,220 MB exceeds the largest TCP window size defined in RFC 7323 [RFC_7323], which states that with 14 bits of window scaling for a 64 KB window, the maximum window size is then 1 GB.

The SWITCH tool also assumes a single stream, so while for a given window size the SWITCH tool might tell us that the maximum throughput achievable is, say, 20 Gbit/s, in practice multiple streams of data can be sent in parallel. Still, such an estimate is useful because it tells us that if we can use four parallel streams, the capacity is available, and the kernel scales properly, we should get close to 80 Gbit/s.

3.5 MTU Size

The Maximum Transmission Unit (MTU) of a path can have a significant impact on performance. Most Ethernet LANs run at a standard 1500 MTU, but larger frames can be beneficial, particularly for large-scale data transfers. This is because fewer frames place less demand on network device CPUs, and the larger frames allow a quicker TCP ramp up or recovery from a loss event due to the packets being up to six times as big [ESnet_Fasterdata_MTU].

GÉANT and many NRENs, including SURF and RNP, support 9000 MTU on their backbones, but for transfer nodes to benefit from this, all systems and links on the path need to support the larger MTU.

One of the important considerations here, particularly for IPv6 where only end hosts can perform fragmentation and thus should run Path MTU Discovery, is that no devices on the path should filter ICMP packet too big (PTB) messages, otherwise senders cannot adjust their sending frame size appropriately. Such filtering is discussed in RFC 4890 [RFC_4890].

Before running tests over the BELLA link, some preliminary experiments were performed to gauge the impact of MTU on data transfers. This question was somewhat academic for our BELLA tests as the end-to-end path already supported the higher 9000 MTU, but there are many sites running at 1500 and for whom the question of potential performance gain should be an interesting one.

We looked at the results of a daily test involving three perfSONAR hosts: Jisc perfSONAR in London (ps-london), Brookhaven National Laboratory (BNL) in the USA, and Imperial College in London, UK (Imperial). BNL has a fixed MTU of 9000, while Imperial uses 1500. We set the MTU at ps-london to 1500 and then 9000. The tests from London to BNL with one-way delay (OWD) of approximately 30 ms show up to a 40% increase in throughput compared to tests to Imperial if an MTU of 9000 is used in ps-london, as shown in Table 3.1.

ps-london	to BNL (Gbit/s)	from BNL (Gbit/s)	to Imperial (Gbit/s)	from Imperial (Gbit/s)
1500	5.86	6.67	12.35	10.1
9000	13.6	15.1	12.8	10.6

Table 3.1: Testing from BNL and Imperial to London (OWD 3 ms) and BNL (OWD 30 ms). Note the Imperial figures for 9000 MTU from ps-london are (approximately) the same, as Path MTU Discovery will have negotiated the MTU down to 1500.

We were curious as to how much a larger 9000 MTU might improve performance for tests run over the BELLA link when compared to a more standard 1500 MTU.

As stated above, the MTU tests did not set the kernel MTU, rather *pscheduler* was used with the `-m (--mss)` option to effectively reduce the MTU; in practice, the MTU would be the TCP MSS plus the IP header.

Figure 3.1 and Figure 3.2 show the effect of varying the MTU between 1500 and 9000 on the throughput achieved, as plotted by the SURF streaming telemetry system.

Figure 3.1 shows *iperf2* for 9000 MTU (left) and 1500 MTU (right), while Figure 3.2 shows the same for *iperf3*. In each case H-TCP was used with 4 streams and 256 MB windows. The throughput with 9000 MTU was roughly 50% greater in each case.

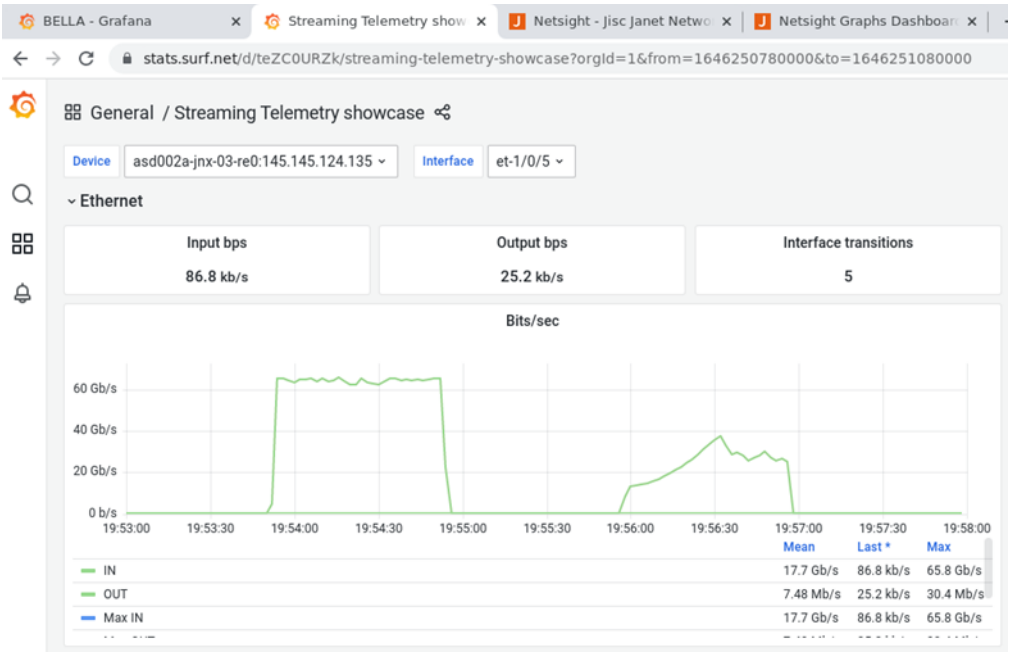

Figure 3.1: *iperf2*, with 9000 MTU (left) and 1500 MTU (right)

Figure 3.2: *iperf3*, with 9000 MTU (left) and 1500 MTU (right)

3.6 Packet Pacing

Packet pacing can have a positive effect on data flows and transfers. The default advice on Fasterdata is to always use fair queuing (*fq*) on Linux. This can be configured alongside the TCP buffer sizes and congestion algorithm within the system configuration [ESnet_Fasterdata_Linux]. Tests run by Brian Tierney of ESnet in 2016 demonstrate the potential value of pacing [ESNet_Tierney].

As described above, it is possible to set the maximum rate at which flows can be sent when running *iperf3* via *pscheduler* with the `-b (-- bandwidth)` option. We ran some tests of this option, and confirmed the pacing for that scenario is per stream. We found that on the systems we used the pacing scaled nicely up to around 8 Gbit/s, but that above this value rates started to become limited, so we were unable to drive single streams above 20 Gbit/s over this link.

We chose not to use pacing in the tests we report below, but it is certainly a topic worth exploring further in the future. It would be timely to look at rerunning the type of tests reported by ESnet above with current systems and configurations.

4 Results of Throughput Tests over BELLA

In this section we report the throughput results achieved between the SURF and RNP servers over the BELLA link while varying certain parameters. The traffic throughput for each test is presented through the SURF streaming telemetry plot with accompanying notes. The test duration was typically 30–60 seconds, and thus there will be activity in the plot, typically a peak, corresponding to each test.

In all our tests, the systems in SURF and RNP and along the full path were running at 9000 MTU.

Given the SURF system was IPv4 only, the tests all used IPv4, but there should be very little performance difference between IPv4 and IPv6. Further experiments could verify this.

4.1 Initial BELLA Experiments

4.1.1 Varying the Congestion Algorithm with 4 Streams and a 128 MB Window

The TCP algorithms used in the first set of tests were Reno, H-TCP and CUBIC, with four streams and a 128 MB window size.

The first three tests in the top-left panel of Figure 4.1 show results using *iperf3* for Reno, H-TCP and CUBIC respectively; these each reach 20 Gbit/s. The second group of three tests for the same algorithms show *iperf2* throughput is higher, with peaks at up to 34.2 Gbit/s.

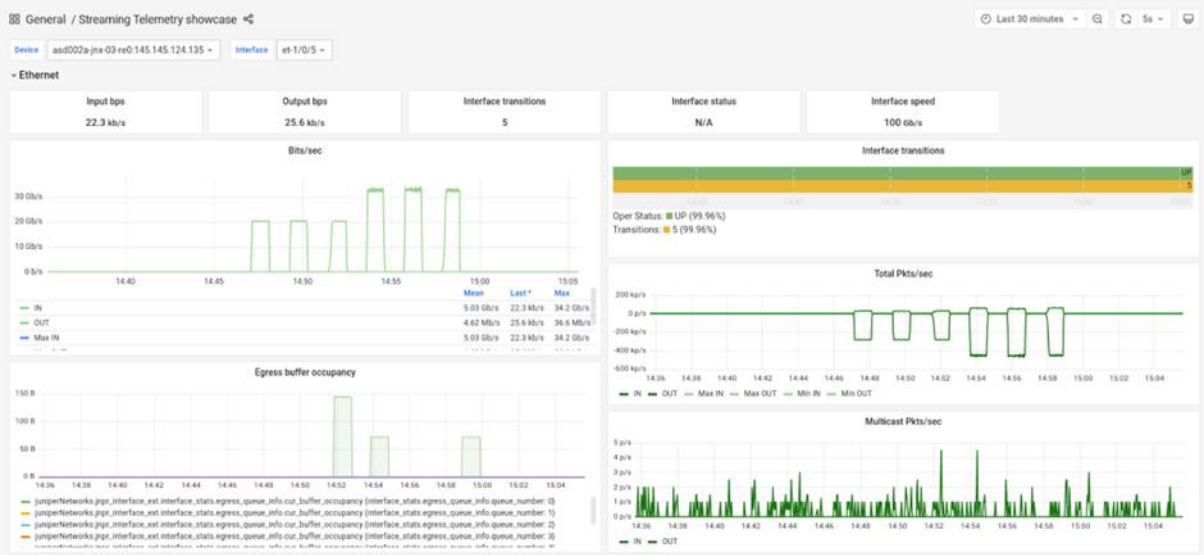

Figure 4.1: The difference in throughput between *iperf3* (top-left panel left, with Reno, H-TCP, CUBIC) and *iperf2* (top-left panel right, with Reno, H-TCP, CUBIC)

4.1.2 Varying the Number of Streams with H-TCP and a 128 MB Window

In the next set of tests, H-TCP was used with a 128 MB window, while the number of streams used was varied.

In Figure 4.2 the first five results in the top-left panel show results with *iperf3* and 1, 2, 4, 6 and 8 streams, with increasing, but not linearly increasing, throughput. The second group of five tests used *iperf2*, again with 1, 2, 4, 6 and 8 streams.

The *iperf2* results show steady growth towards a larger throughput, up to 72 Gbit/s, with results scaling with the number of streams.

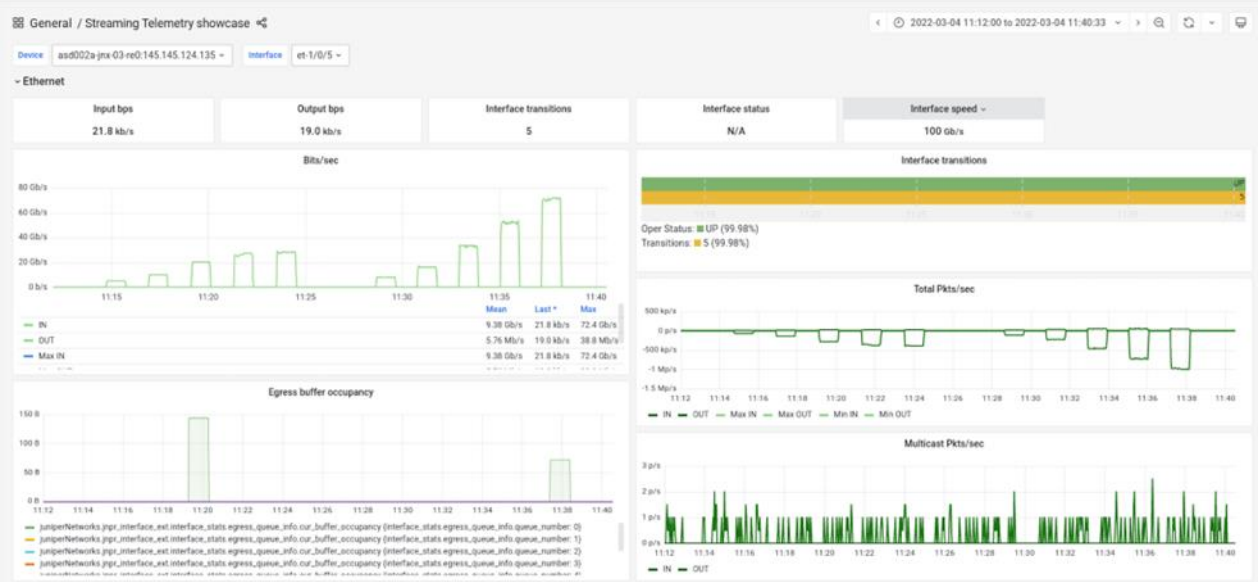

Figure 4.2: The effect of the number of streams (1, 2, 4, 6 and 8) and *iperf* version (top-left panel left, *iperf3*; top-left panel right, *iperf2*) on throughput

For *iperf3* the scaling is poorer for the reasons described earlier. A gain was observed when the number of streams was increased from 4 to 6, but no gain was observed going from 6 to 8 streams. All the *iperf3* tests in Figure 4.2 achieved less than 35 Gbit/s; this is expected and is documented as a feature of the *iperf3* software, which aims at maximising throughput on one stream [ESnet_Fasterdata_iperf].

4.1.3 Exploring a Larger 256 MB Window with 6 and 8 Streams

In the next set of tests, we fixed the window size at 256 MB. In Figure 4.3 the first six tests show results using *iperf3*, which reach at most 30 Gbit/s, whilst the second group of six tests using *iperf2* show increased throughput up to 93.2 Gbit/s.

For each set of six tests the first two are Reno with 6 and 8 streams, the second pair are H-TCP with 6 and 8 streams, and finally the last two are CUBIC with 6 and 8 streams.

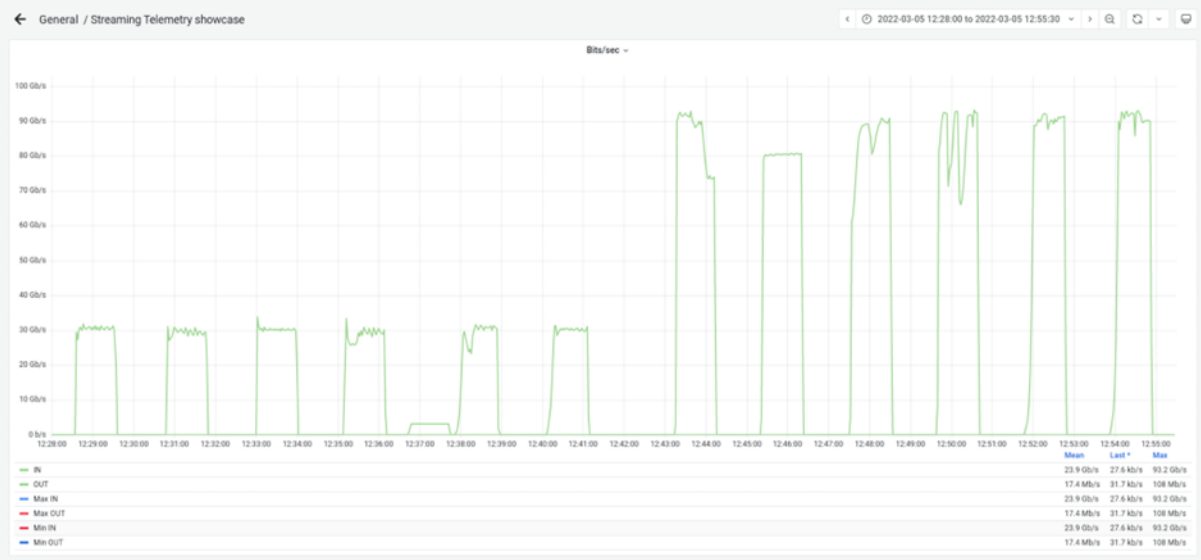

Figure 4.3: The difference in throughput between *iperf3* (left) and *iperf2* (right) using different congestion algorithms, 6 and 8 streams, and window size 256 MB

The tests show the best results that could be achieved for *iperf3* with the SURF server, which had 4 cores and 8 hyper-threads, though as we see below, the system was still performant enough to fill the link using *iperf2* and optimal tuning.

4.2 Achieving Maximum Throughput

The above tests showed that throughput of just over 90 Gbit/s was possible with a 256 MB window and six streams. In this section we report the steps to achieve maximum throughput in tests over the BELLA link.

Maximising throughput can be seen as a combinatorial optimisation problem involving maximum link capacity, round-trip time, the limits of the hardware (including CPU and cores) and operating system, and the configurations of the TCP stacks in the hosts involved, in particular the sizes of windows used in the test, the path MTU, and the number of parallel streams.

The BDP can help to configure sizes of windows: it says how much data can be transmitted and in flight before it is acknowledged. Tools such as the SWITCH TCP throughput calculator can help us to get an idea of the ideal window size for an ideal link. However, as described above, it is unaware of limits in the hardware and operating systems, and assumes a single stream.

Given an estimate of performance for a single stream, it is important to find out how many streams we can send in parallel that will scale before hitting concurrency limits in the kernel. By steadily increasing the window sizes and number of streams used, and observing the presence of retransmissions, it was determined that a window size of 512 MB, with 6 streams, should allow the link to be filled.

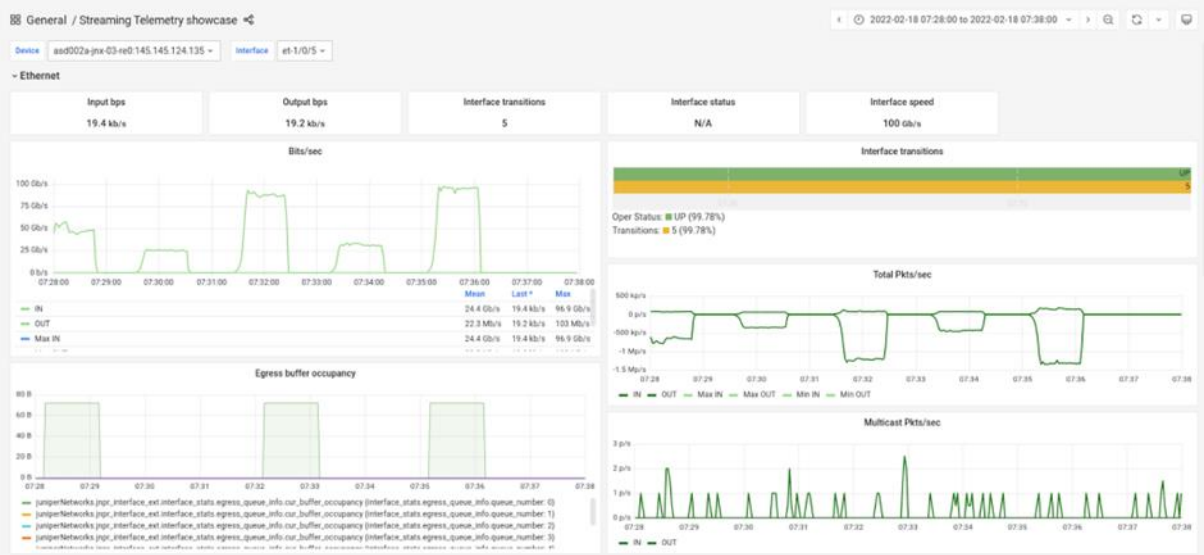

Figure 4.4: Maximum throughput was achieved using CUBIC and 512 MB windows; here the middle peak in the top-left panel is H-TCP with *iperf2*, and the right-hand peak is CUBIC with *iperf2*

Having established the parameters to use, Figure 4.4 shows that a maximum throughput of 96.9 Gbit/s was achieved at 07:36 using *iperf2* with CUBIC, six streams and 512 MB windows. No performance increase was obtained with eight streams. It is noticeable in Figure 4.4 how flat the curve is at the maximum throughput.

4.3 Future Work

As we ran the experiments, we identified various strands of future work that could be performed, which includes:

- **IPv6** – Repeating the tests using IPv6. We do not expect significantly different results, but given the increasing prominence of IPv6 (currently around 40% of Internet user traffic, for example as reported by Google [Google_IPv6_Stats]), it is important that its performance on R&E infrastructures is verified.

- **Packet pacing** – The tests reported by ESnet in 2016 were very interesting, and it would be timely to repeat these on current systems and configurations. For example, can the performance be improved? Does pacing make traffic fairer to other applications?

- **TCP-BBRv2** – Early versions are available, but tests once BBRv2 is in mainstream distributions and more readily available to users would be very useful. How does it perform? What is the impact on flows using other TCP congestion algorithms?

- **Streaming telemetry** – Having access to fine-grained detail of traffic data was very useful while running the tests. While not a performance tool, there is clear potential for R&E networks to deploy streaming telemetry to gain better insight into traffic behaviour. Further deployments would be useful.

- **Test tool reporting accuracy** – We saw some examples where the test tools over-reported the results achieved. Rerunning tests and cross-checking with (for example) *ethtool* would be a useful exercise.

The authors would welcome input from or collaboration with the R&E community in these areas.

5 Conclusions

Our experiments set a target of maximising traffic throughput over the R&E network path between SURF in Europe and RNP in Brazil. The task required that we find an optimal TCP tuning for the hosts at SURF and Fortaleza.

The perfSONAR toolkit was well-suited to running the experiments; we could initiate throughput tests from SURF to RNP using *pscheduler* on a remote perfSONAR server on the Janet network, and use *pscheduler* options to request different window sizes, congestion algorithms, number of streams, and TCP MSS (related to the MTU).

The SURF streaming telemetry platform was also very useful for viewing fine-grained detail of each short-duration (30 to 60 seconds) test; this would not be possible using a traditional SNMP-based monitoring system with 5-minute traffic averages.

Although the SURF to RNP network path was 9000 MTU throughout, tests were also performed to look at the effect of changing the MTU size. For the path which was just over 100 ms RTT, our tests showed a 50% benefit with using 9000 MTU.

The best throughput achieved was 96.9 Gbit/s, as shown by SURF's streaming telemetry system in Figure 4.4. For this, CUBIC was used, with 512 MB windows and six parallel streams. The peak throughput will of course depend on other traffic on the link; the tests were typically run for short 30–60 second bursts to minimise impact on any production traffic.

We also tested and reported on the effectiveness of different congestion algorithms: Reno, H-TCP and CUBIC. Plots for our BELLA scenario generally showed CUBIC as the best for throughput, but also showed good stability in Reno. As noted in the future work section above, it would be interesting to use the newer TCP-BBRv2 for comparison, once it is part of standard Linux distributions.

In practice, it is the application performance that matters when transferring large volumes of data, and different applications use different approaches, with varying numbers of concurrent transfers and parallel streams, for files and data sets with different profiles.

The tests reported here show that the 100G BELLA link can be utilised to almost its maximum capacity with a small number (six) of *iperf2* TCP flows for a given set of tuning parameters, and therefore in practice there should be no reason why real data transfers should not achieve similar performance, subject to other traffic on the link.

References

[BELLA] https://bella-programme.redclara.net/index.php/en/

[BELLA_Expansion] https://connect.geant.org/2022/01/21/join-the-inauguration-of-the-bella-connectivity-between-brazil-argentina-and-chile-26-january-2022

[BELLA_Launch] https://connect.geant.org/2021/11/23/bella-connectivity-brings-benefits-to-high-energy-physics-research

[BELLA_Map] https://bella-programme.redclara.net/images/slider/bella_map_website_banner.png

[BRIAN_BELLA] https://public-brian.geant.org/d/Kk8ARPInk/bella?orgId=5

[CUBIC] https://www.cs.princeton.edu/courses/archive/fall16/cos561/papers/Cubic08.pdf

[Dordal] Peter L. Dordal, "An Introduction to Computer Networks" https://intronetworks.cs.luc.edu/1/html/index.html

[ESnet_BBRv2] https://indico.geant.org/event/1/contributions/42/attachments/52/66/20220616-dart-BBRv2-v2.pdf

[ESnet_Fasterdata] https://fasterdata.es.net/

[ESnet_Fasterdata_iperf] https://fasterdata.es.net/performance-testing/network-troubleshooting-tools/iperf/

[ESnet_Fasterdata_Linux] https://fasterdata.es.net/host-tuning/linux/

[ESnet_Fasterdata_MTU] https://fasterdata.es.net/network-tuning/mtu-issues/

[ESnet_Fasterdata_pscheduler] https://fasterdata.es.net/performance-testing/network-troubleshooting-tools/pscheduler/

[ESnet_Tierney] Brian Tierney, "Advantages of TCP pacing using FQ" https://fasterdata.es.net/assets/fasterdata/FQ-pacing-results.pdf

[ethr] https://github.com/microsoft/ethr/blob/master/README.md

[Google_IPv6_Stats] https://www.google.com/intl/en/ipv6/statistics.html

[GridFTP] https://ogf.org/documents/GFD.20.pdf

[H-TCP] https://www.hamilton.ie/net/htcp3.pdf

[iperf2] https://sourceforge.net/projects/iperf2/

[iperf3] https://iperf.fr/iperf-download.php

[Mathis] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, Teunis OTT, "The macroscopic behavior of the TCP congestion avoidance algorithm". *ACM SIGCOMM Computer Communication Review*, Volume 27, Issue 3, July 1997, pp. 67–82 https://dl.acm.org/doi/10.1145/263932.264023

[perfSONAR] https://www.perfsonar.net/

[perfSONAR_Throughput] https://docs.perfsonar.net/pscheduler_ref_tests_tools.html#throughput

[PMW_Mar2022] https://events.geant.org/event/1084/

[Reno] https://www.rfc-editor.org/rfc/rfc5681
[RFC_4890] https://www.rfc-editor.org/rfc/rfc4890
[RFC_7323] https://www.rfc-editor.org/rfc/rfc7323.html
[SWITCH_TCP_Throughput] https://www.switch.ch/network/tools/tcp_throughput/
[T&DW_Apr2022] https://events.geant.org/event/1104/
[TCP_BBRv2] https://www.ietf.org/proceedings/106/slides/slides-106-iccrg-update-on-bbrv2-00

Glossary

BDP	Bandwidth-Delay Product
BELLA	Building the Europe Link to Latin America
BNL	Brookhaven National Laboratory
BRIAN	Backbone Router Interface Analytics
CERN	European Organisation for Nuclear Research
CPU	Central Processing Unit
DTN	Data Transfer Node
FQ	Fair Queuing
H-TCP	A TCP variant that is suitable for deployment in high-speed and long-distance networks, as well as conventional networks
ICMP	Internet Control Message Protocol
IP	Internet Protocol
LAN	Local Area Network
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NIC	Network Interface Card
NREN	National Research and Education Network
OWD	One-Way Delay
PTB	Packet Too Big
R&E	Research and Education
RAM	Random Access Memory
RTT	Round-Trip Time
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TNC	The Networking Conference
WLCG	Worldwide Large Hadron Collider Computing Grid
WP	Work Package
WP6	GN4-3 Work Package 6 Network Technologies and Services Development
WP6 T3	WP6 Task 3 Monitoring and Management