Sliced WANs for Data-Intensive Science: Deployment Experiences and Performance Analysis

Edgard C. Pontes[1], Vitor Zanotelli[1], Magnos Martinello[1], Jordi Ros-Giralt[2], Everson S. Borges [1], Moisés R. N. Ribeiro[1] and Harvey Newman[3]

[1] Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, 514, Goiabeiras – 29.075-910 – Vitória – ES – Brasil

[2]Qualcomm Europe, Inc –

[3]Caltech, California Institute of Technology

Abstract. *The task of transferring massive data sets in Data-Intensive Science (DIS) systems, such as those generated from high energy experiments at CERN in Switzerland and France, and at the Sirius synchrotron light source in Brazil, often rely on physical WAN infrastructure for network connectivity that is provided by various National Research and Education Networks (NRENs), including ESnet, Géant, Internet2, RNP, among others. Sliced WANs bring a new paradigm for infrastructure yet to be exploited by DIS, but a realistic study of these particular systems poses a significant challenge due to their complexity, scale, and the number of factors affecting the data transport. In this paper, we address some of these challenges by deploying and evaluating a virtual infrastructure for data transport within a representative national-scale WAN. Our approach here encompasses two main aspects: i) Evaluating the performance of TCP congestion control algorithms (BBR versus Cubic) when only a single path is available for the data transfer; and ii) Assessing the performance of flow completion times (related to the management of bandwidth allocation) for sets of interdependent transfers in an environment provided by a network slice.*

Resumo. *A tarefa de transferir conjuntos massivos de dados em sistemas Data-Intensive Science (DIS), como aqueles gerados a partir de experimentos de alta energia no CERN na UE e na fonte de luz síncrotron Sirius no Brasil, muitas vezes depende de infraestrutura WAN física para conectividade de rede que é fornecido por diversas Redes Nacionais de Pesquisa e Ensino, incluindo ESnet, Géant, Internet2, RNP, entre outras. As WANs fatiadas trazem um novo paradigma para a infraestrutura ainda a ser explorada pelo DIS, mas um estudo realista desses sistemas específicos representa um desafio significativo devido à sua complexidade, escala e ao número de fatores que afetam o transporte de dados. Neste artigo, damos os primeiros passos para enfrentar esses desafios, implantando e avaliando uma infraestrutura virtual para transporte de dados dentro de uma WAN representativa em escala nacional. Nossa abordagem aqui abrange dois aspectos principais: i) Avaliar o desempenho de algoritmos de controle de congestionamento TCP (BBR versus Cubic) quando apenas um único caminho está disponível para a transferência de dados; e ii) Avaliar o desempenho dos tempos de conclusão dos fluxos (relacionados com a gestão da alocação de largura de banda) para conjuntos de ambientes de transferências interdependentes fornecidos por uma fatia de rede.*

1. Introduction

Data-intensive science (DIS) is gaining increasing importance as it drives the exploration of fundamental scientific questions by collecting and processing vast amounts of scientific data [Dunefsky et al. 2022]. In the context of large-scale data intensive sciences, scientists collaborate on various projects to design infrastructure for data generation, establish data centers for storage and processing, and develop a network infrastructure for efficient data transfer. An exemplary instance is the global system implemented by CERN and its collaborating partners, featuring 170 data centers distributed across 40 countries [Newman et al. 2003].

One could suppose that crafting the transfer of massive data sets would be a straightforward challenge by using existing widely available facilities and techniques developed by the networking community. Unfortunately, the type of data transport in DIS introduces unique challenges that existing techniques do not tackle adequately. Although TCP and the associated congestion control algorithms are capable of achieving reasonably fair sharing among flows, in DIS this task needs to be coordinated with the management of bandwidth allocation for sets of interrelated transfers, such as those associated with the same scientific experiments. Additionally, it often involves making the appropriate selection of a TCP congestion control algorithm and also intricate efforts in tuning TCP parameters to improve performance [Weigle and chun Feng 2002].

DIS applications could also exploit traffic engineering and manage constraints related to the network topology in the interest of transporting their massive data sets. But, unfortunately, regular network operators usually do not allow access to multiple path configuration. In addition, the control over link features in that topology remains out of reach of the DIS community when trying to match transport requirements of a given scientific collaboration. On the other hand, softwarization of network functions, combined with massive virtualization of computing resources, are being used to create the network slicing paradigm. This enabling technology has the potential of creating overlaying solutions to the DIS community that are yet to be fully investigated. In this direction, general purpose experimental slice-based facilities are recently being made available, and we believe that DIS could largely benefit from them. However, finding out more efficient ways to exploit traffic engineering, considering now full access to configurations of network nodes and their interfaces (including shared and programmable NICs), is an intricate and timely cross-layer issue in need to be addressed by DIS.

The network shown in **Fig. 1** is from the FABRIC Testbed[1], which serves our purposes as a WAN topology that represents a national scale DIS network, where multipath transport can potentially be effectively exploited. This facility allows us to perform a realistic study of a sliced infrastructure designed to meet the data transport of DIS systems, while simultaneously addressing their complexity, scale, and a number of factors affecting the data transport. In this paper, our goal is to investigate the interdependence arising in DIS to help architects formulate request sliced resources together with congestion control to meet the transport needs. The experimental investigation here scales down DIS data volumes and transmission rates accordingly for building a prototype[2] representation that yields the following contributions:

[1] `https://portal.fabric-testbed.net/`

[2] Our definition of prototype slice is scaling down the transport volumes present in DIS.

- **i)** We evaluate the performance of TCP congestion control algorithms (BBR versus Cubic) and the use of multiple simultaneous TCP connections to transfer data, when only a single path is available for the data transfer.
- **ii)** We assess the performance of flow completion times on a multi path slice which is a WAN deployed within a national-scale testbed. Our focus is on the management of bandwidth allocation for sets of interdependent transfers, conducted within a virtual environment provided by a network slice;

The remainder of this paper is as follows. **Section 2** critically assesses experimental testbeds. In **Section 3**, we present specific FABRIC features to support slice definitions, including booking a set of components, configuration deployment, and setup automation. **Section 4** brings the experimental evaluation results assessing data transport efficiency for DIS. Conclusions are drawn, and future work is discussed in **Section 5**.

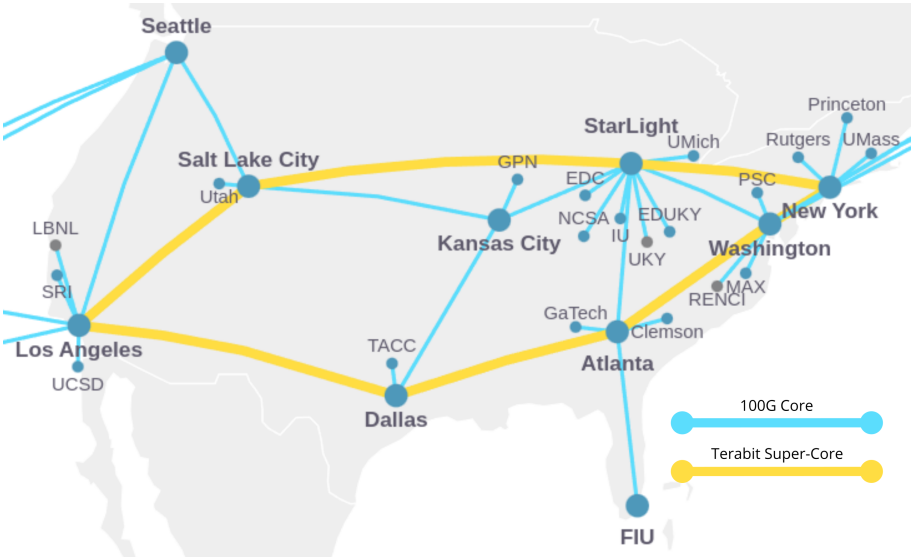

Figure 1. FABRIC nationwide network topology

2. Slicing Networks in Experimental Testbeds for Data-Intensive Science

Looking at the state-of-the-art experimental testbeds, there have been many projects that have provided experimental facilities for networking experimentation [Berman et al. 2014, M. et al. 2014, Salmito et al. 2014, Both et al. 2019]. The GENI project was the pioneer in building an open and distributed infrastructure at scale for research and education [Berman et al. 2014]. The OFELIA project [M. et al. 2014] created an experimental facility that was based on SDN/OpenFlow to control the network environment. With a similar goal, Japan had the RISE project, which was an OpenFlow-based research infrastructure for large-scale network experiments on the Japan Gigabit Network [Huang et al. 2017].

In Brazil, the FIBRE project [Salmito et al. 2014] built federated testbeds to provide a large-scale research platform. Furthermore, the FUTEBOL project [Both et al. 2019] was conceived to enable experimental research on optical, wireless, and cloud convergence that has been an enabler of the recent 5G networks architecture[3]. Currently, a comprehensive initiative called slicing future Internet infras-

[3]https://porvir-5g-project.github.io

tructures (SFI2) gathers efforts toward designing multi-domain and multi-technology integrated experimental network deployment. SFI2 main features are native machine learning optimization, energy-efficient aware slicing, and slicing-tailored security functionalities for the practical domain [Martins et al. 2023].

The FABRIC testbed, as introduced in [Baldin et al. 2019], is designed to function as a national research infrastructure, facilitating cutting-edge and exploratory research on a large-scale network. FABRIC is characterized by a deep programmability model (utilizing P4, SmartNiCs, and DPDK nodes), with terabits-per-second core network nodes that interconnect existing facilities such as GENI [Berman et al. 2014], as well as future resources such as Open Cloud [Zink et al. 2021], campus networks, clusters, national high-performance computing (HPC) and data facilities.

Unlike previous efforts, FABRIC specifically aligns with the requirements of massive data transfer by offering a typical slice created on a wide-area network at the national level. This slice can be connected using user-specified networking services, enabling multiple configurations not presently available in the previous testbeds. For example, the slice model allows us to tune virtual infrastructure toward meeting particular requirements from DIS transport. Setting the capacity of the virtual interfaces evidently impacts flow completion time; besides the traditional efforts to look into buffer sizes and the values of RTT to evaluate the performance of TCP congestion control algorithms (e.g., BBR, Cubic, Reno) in DIS.

A second challenge on the data transport performance is related to how the routers use packet buffers to prevent packet loss during congestion. Proper sizing is crucial; if too small, it causes high loss and under-utilization, while if too large, it leads to unnecessary delays. Vendors tend to oversize buffers, and operators often configure larger buffers than needed, resulting in increased costs and delays [Spang et al. 2022]. Also, BBR is a congestion control algorithm that is gaining popularity, and addressing the bufferbloat issue in traditional loss-based algorithms by preventing excessive queue buildup while maintaining high throughput [Vargas et al. 2021]. However, research shows that BBR can lead to poor performance with deep buffers, causing inflated latencies and degraded quality [Cao et al. 2019].

In contrast to prior efforts that primarily concentrated on tuning TCP itself, e.g., [Weigle and chun Feng 2002] and [Cao et al. 2019], our approach involves **tuning a slice** with one or multiple TCP flows transporting data across a WAN topology over a single or even multiple paths. Different than previous works, our method enables the deployment and performance analysis based on a slice as a prototyping process, providing insights to better understand how DIS tranport applications can benefit from sliced-WANs.

3. Slice Deployment and Experimentation with the FABRIC Testbed

To perform a realistic study of the data transport efficiency of DIS systems [Dunefsky et al. 2022], a slice representing a large-scale WAN was requested to be created on the FABRIC Testbed. After the deployment phase of the booked slice, there is the configuration and automation phases, whereas the latter makes experiments easily repeatable when reaching the final phase, which is focused on collecting results.

Booking a slice: An example of how to select nodes and link resources to book a slice using FABlib library is shown in **Code 1**. This slice is composed of two L2 overlay

networks (lines 13 and 14), and the subnets (lines 6–7) attached to these connections (lines 19–21 and 26–31). Selection and addition (lines 18 and 25) of NIC model NIC_ConnectX_5 to the slice[4]. As a result, we can visualize the requested slice in **Fig. 2**. All configuration files in this Section are made available at a GitHub[5].

```
1  # Creates a new slice
2  slice = fablib.new_slice(name=slice_name)
3
4  # Creates the subnetworks
5  subnet1 = IPv4Network("192.168.1.0/24")
6  subnet2 = IPv4Network("192.168.2.0/24")
7
8  # Lists of available ips
9  net1_available_ips = list(subnet1)[1:]
10 net2_available_ips = list(subnet2)[1:]
11
12 # Creates l2 overlay networks
13 net1 = slice.add_l2network(name='net1', subnet=subnet1)
14 net2 = slice.add_l2network(name='net2', subnet=subnet2)
15
16 # Adds the H1 node to the slice
17 h1 = slice.add_node(name='h1', site='SEAT', image=os_name)
18 [h1n1, h1n2] = h1.add_component(model=model_name, name='nic1').get_interfaces()
19 h1n1.set_mode('config')
20 net1.add_interface(h1n1)
21 h1n1.set_ip_addr(net1_available_ips.pop(0))
22
23 # Adds the R1 node to the slice
24 r1 = slice.add_node(name='r1', site='SALT', image=os_name)
25 [r1n1, r1n2] = r1.add_component(model=model_name, name='nic1').get_interfaces()
26 r1n1.set_mode('config')
27 net1.add_interface(r1n1)
28 r1n1.set_ip_addr(net1_available_ips.pop(0))
29 r1n2.set_mode('config')
30 net2.add_interface(r1n2)
31 r1n2.set_ip_addr(net2_available_ips.pop(0))
```

Code 1: Example of how to book a slice using FABlib library

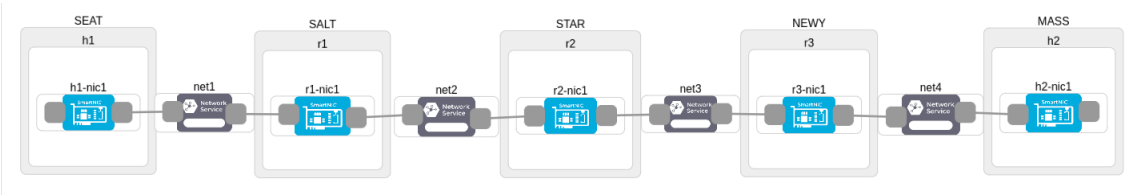

Figure 2. Slice visualization at Fabric Testbed

Configuration and automation: The allocated nodes are virtual machines (VMs) with Linux OS attached to network interfaces. They need to turned into routers or be converted into end devices, which in DIS they are called data transfer nodes (DTNs). For example, the configuration to apply CPU Pinning and Non-Uniform Memory Access (NUMA) is especially interesting in systems with multiple processors or sockets, on each slice node is presented in **Code 2**. This enforces the allocation of all virtual CPUs (vCPUs) assigned to the VM on the specific NUMA node containing the relevant components (e.g. network interfaces). In a VM, aligning vCPUs and memory with the physical NUMA nodes helps reduce the time it takes for processors to access the memory

[4]Each NIC model **NIC_ConnectX_5** added to the slice corresponds to two node interfaces.
[5]https://github.com/nerds-ufes/sbrc2024

associated with their node. Note that a system restart across all nodes is required for the configuration changes to be implemented.

```
slice = fablib.get_slice(name = slice_name)

for node in slice.get_nodes():
    # Pin all vCPUs for VM to same Numa node as the component
    node.pin_cpu(component_name='nic1')

    # Pin memmory for VM to same Numa node as the components
    node.numa_tune()

    # Reboot the VM
    node.os_reboot()
```

Code 2: Example of how to apply performance tuning on all nodes using FABlib library

Experiment Automation: the **Code 3** is an example of automation process that allows us to configure a large number of system parameters such as: the data size to be transferred, the target IP address of the iPerf3 server (line `1`), the selection of TCP algorithms (line `2`), the added delays (line `3`), the buffer size assigned to routers (line `4`), and bandwidth limit (`5`). The TC-NetEm and TC-tbf parameters are configured on router `R1` (lines `7-12`) and changed at each iteration of the iPerf3 flows (lines `18` and `21`). Note that congestion control algorithms are here modified only on hosts/DTNs (lines `15` and `16`).

Running the experiments and collecting data: All tests were performed in the FABRIC Testbed using the Jupyter Notebook, provided by JupyterLab (v. 3.4.6) and the FABlib library (v. 1.6.0). Each node in the experiments was equipped with the default configuration of the testbed, which are 2 cores, 8 GB of RAM, and 10 GB of disk space. The operating system used for the tests was Rocky Linux 8.9 (Green Obsidian), Linux Kernel 4.18.0, while the interfaces used were dedicated NICs "Mellanox ConnectX-5 Dual Port 10/25GbE". Traffic generation and throughput measurements were performed using iPerf3 (v. 3.5.0).

4. Assessing Data Transport Efficiency for DIS

Suppose that our goal is to transfer a certain amount of data in a WAN so that all the flows representing the interdependent transfers must complete in the shortest time possible. This is a common application in DIS networks, where users need to share data obtained from their experiments, often involving terabytes (or even petabytes) of information, with collaborators around the globe. So, we want to investigate the data transport efficiency by evaluating the performance of a set of flows that are being transferred on a WAN deployed within a national-scale testbed.

4.1. Data transfer performance over a coast-to-coast single path

The first slice created is to investigate the performance of the TCP congestion control and the use of multiple TCP streams to transfer data. The slice sets a coast-to-coast linear topology with 5 nodes (`H1`, `R1`, `R2`, `R3`, and `H2`) deployed, respectively, geographically located at the following sites: `SEAT` (Seattle), `SALT` (Salt Lake City), `STAR` (StarLight), `NEWY` (New York) and `MASS` (UMass), as shown in **Fig. 3**.

With this simple setup, we can already investigate how to tune the slice network parameters comparing BBR vs. Cubic in terms of throughput and flow-completion time.

```
size, target = '1G', '192.168.4.2'
algs = ['cubic','bbr']
delays = ['0', '60', '120', '180', '240', '320', '380', '440']
buffers = ['100K', '250K', '500K', '1M','5M', '10M', '25M', '50M']
rates = ['300']

r1.execute(f'sudo tc qdisc add dev eth1 root handle 1: htb default 10;\
             sudo tc class add dev eth1 parent 1: classid 1:10 htb rate {rates[0]}Mbit;\
             sudo tc qdisc add dev eth1 parent 1:10 netem delay 0ms;\
             sudo tc qdisc add dev eth2 root handle 1: htb default 10;\
             sudo tc class add dev eth2 parent 1: classid 1:10 htb rate {rates[0]}Mbit;\
             sudo tc qdisc add dev eth2 parent 1:10 tbf rate {rates[0]}Mbit buffer
             ↪  {rates[0]} limit {rates[0]}')
h2.execute(f'iperf3 -s -p 5201 -i 2 -D')
for alg in algs:
    h1.execute(f'sudo sysctl -w net.ipv4.tcp_congestion_control={alg}', quiet=True)
    h2.execute(f'sudo sysctl -w net.ipv4.tcp_congestion_control={alg}', quiet=True)
    for delay in delays:
        r1.execute(f'sudo tc qdisc change dev eth1 parent 1:10 netem delay {delay}ms')
        for buffer in buffers:
            for rate in rates:
                r1.execute(f'sudo tc qdisc change dev eth2 parent 1:10 tbf rate
                ↪  {rate}Mbit buffer {buffer} limit {buffer}')
                h1.execute(f'iperf3 -c {target} -p 5201 -i 2 -n {size} -Z -C {alg} -J |
                ↪  tee > bw-{rate}-sz-{size}-de-{delay}-bf-{buffer}-{alg}.json')
h2.execute(f'sudo pkill iperf3')
r1.execute(f'sudo tc qdisc del dev eth1 root')
r1.execute(f'sudo tc qdisc del dev eth2 root')
```

Code 3: Automation process for setting the multiple configurations including *tc*, data size, buffers size, TCP algorithms and delays.

In addition, multi-thread transport is also investigated in order to understand the effects of multiplexing comparing BBR vs. Cubic. Traffic was generated using iPerf3 at (**Fig. 3** from `H1` host to `H2`), which are the DTNs. All TC-NetEm and TC-tbf configurations were carried out at the router `R1`, more specifically, with the traffic policy applied at the `eth1` and `eth2` interfaces. We used heatmaps to visualize the different network settings. In each heatmap, we show a metric value under different RTT and Buffer size settings. The *tc* tool is used for setting a range of parameters based on values commonly employed in modern networks [Cao et al. 2019]:

- Bandwidth (BW): limited at 300 Mbps per flow (only for single thread experiments);
- File to be transferred: 1GB (for flow-completion time single thread experiments)
- RTT: approximately 90 ms base value plus the following variations: +0, +60, +120, +180, +240, +320, +380 and +440 ms;
- Buffer sizes: 0.1, 0.25, 0.5, 1, 5, 10, 25 and 50 MBytes.

Figure 4 shows the mean of TCP throughput (Left) in Mb/s and the flow completion time (Right) in seconds, for different values of buffer size. We observe that BBR outperforms Cubic, specially for shallow buffer sizes. For instance, under 500 KB buffer size, there are more light reds that indicate lower throughput for Cubic, although the difference is lower for higher values of RTTs (e.g. light colors for RTT greater than 410 ms for both). **Figure 4** (Right) complements this view (i.e. the lower the completion time, the better with light red colors), showing that for lower values of buffer size (under 500 KB), there have been more dark red colors for Cubic, indicating that BBR outperforms Cubic even when RTT is high, for the case of shallow buffer size.

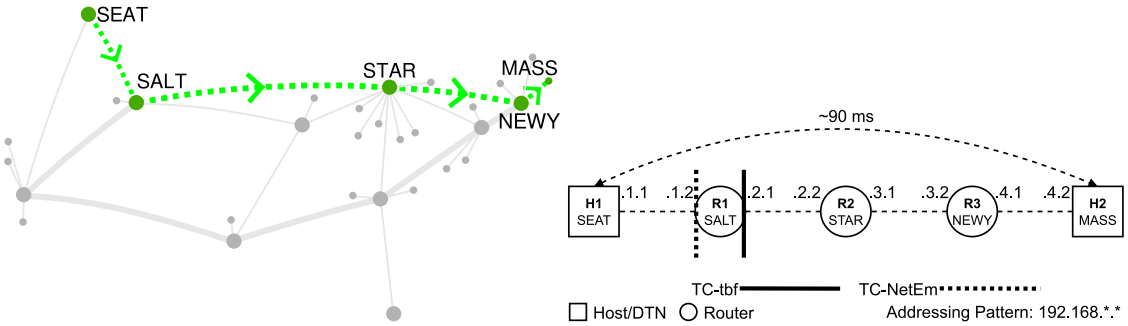

Figure 3. Coast-to-coast WAN deployment in FABRIC Testbed in (Left) with its logical slice linear WAN allocation in (Right).

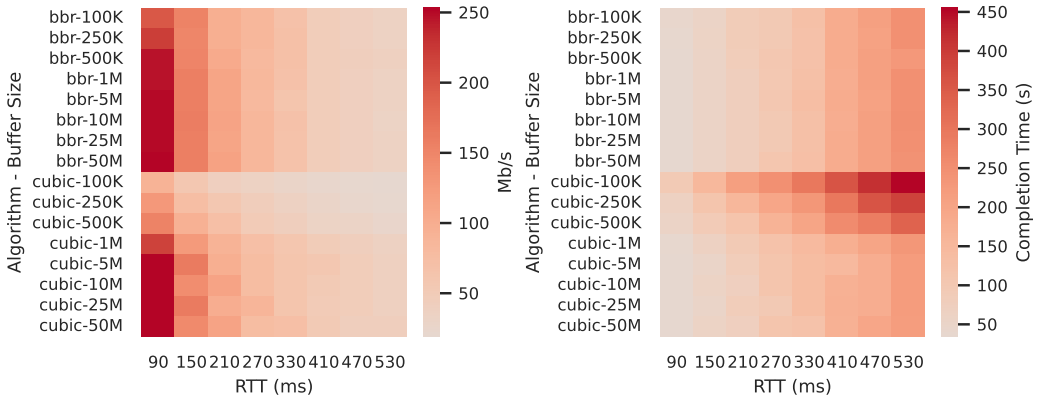

Figure 4. Heatmaps comparing throughput (Left) and flow completion time (Right) for BBR and Cubic algorithms, varying buffer size and delay (RTT).

In order to clarify the differences, we compute the BBR's throughput percentage gain over Cubic as $\frac{BBR-CUBIC}{BBR} \times 100$. **Figure 5** shows the gain (Left) and the number of retransmissions (Right). The red shaded regions and positive values indicate that BBR outperforms Cubic, whereas blue shaded regions and negative values indicate that Cubic outperforms BBR. As we can observe, BBR achieves higher throughput values in shallow buffers (60% better) than Cubic, regardless of the RTT values. However, for deep buffer sizes (beyond 1 Mbyte) and long RTTs (beyond 240 ms), Cubic outperforms BBR (20% better), see the blue colors in the figure.

Figure 5 (Right) shows the number of packet retransmissions for both BBR and Cubic. We observe that BBR often incurs 2 to 6 times more retransmissions than Cubic, see the dark red colors on top of the figure. The key to understanding this result lies in BBR's maintenance of 2 × BDP (Bandwidth-Delay Product) number of packets in flight. When the buffer size is smaller than BDP, BBR will continually have losses [Cao et al. 2019]. Nevertheless, Cubic addresses these losses by substantially reducing its congestion window (cwnd), whereas BBR does not immediately react to the loss (being more aggressive).

The second set of experiments aims to answer a practical question related to the use of multiple TCP streams to transfer data. How much can we enhance on the flow-completion time by utilizing parallel TCP streams? In other words, when we multiplex BBR and Cubic connections, which one will demonstrate superior performance?

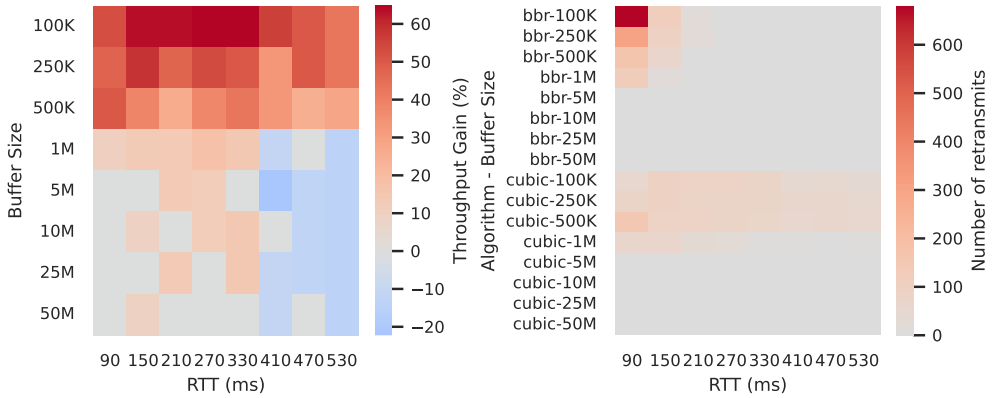

Figure 5. Heatmaps comparing goodput percentage gain (Left) and number of retransmissions (Right) for BBR and Cubic algorithms and varying buffer size and delay (RTT).

Figure 6 shows the time to transfer data of different sizes $1, 5$ and 10 Gbytes. We used multiple TCP streams, varying the number of threads for both BBR and Cubic. As we can see in **Fig. 6** (Left), the flow completion time decreases significantly as the number of threads increases. Moreover, the impact of parallel TCP streams on decreasing the flow completion time is notable up to a specific number of threads (typically showing lower improvement after 15 as indicated by a vertical red line). Interestingly, with an increase in the number of threads, for example beyond 15 in this experiment, Cubic starts to take longer to transmit 5 GB (dotted orange) than BBR to transmit 10 GB (green). Notably, BBR multiplexing appears to be more effective than Cubic. **Figure 6** (Right) focus on the flow completion time difference between BBR vs. Cubic ($Cubic - BBR$) in seconds. While we do observe slight differences favoring BBR, averaging around (15s to 10 GB, 7s to 5 GB, and 1s to 1 GB), the overall effectiveness of increasing the number of threads as a strategy surpasses the impact of selecting TCP algorithms.

Increasing the number of threads has some limitations, though. **Figure 7** (Left) shows the throughput per thread in Mb/s for BBR vs Cubic. As expected, the greater the number of threads, the lower the throughput per thread. However, BBR keeps its performance per thread longer than Cubic. While BBR begins to significantly decline after 20 threads (i.e. throughput aggregated of 5 Gbps achieving the maximum empirical rate), Cubic experiences a performance drop much earlier, at just 5 threads. **Figure 7** (Right) complements the result showing the percentage gain per thread $\frac{BBR-CUBIC}{BBR} \times 100$. The gain is more pronounced from 5 to 20 reaching about 50% in favor of BBR. These results confirm that in scenarios with a high number of threads and a substantial volume of data, BBR multiplexing proves to be more effective than Cubic.

4.2. Data transfer performance over multipath

In the process of allocating a *multipath slice*, network providers commonly offer a network with an uniform path capacity. As a result, a question emerges for DIS applications: In a multipath-sliced network, how flow-completion time can be affected by using different bandwidth values across the paths? Our objective is to empirically demonstrate the existence of a configuration (by finding the bandwidth values in a slice) that both minimizes network completion time and maximizes network throughput. For this purpose,

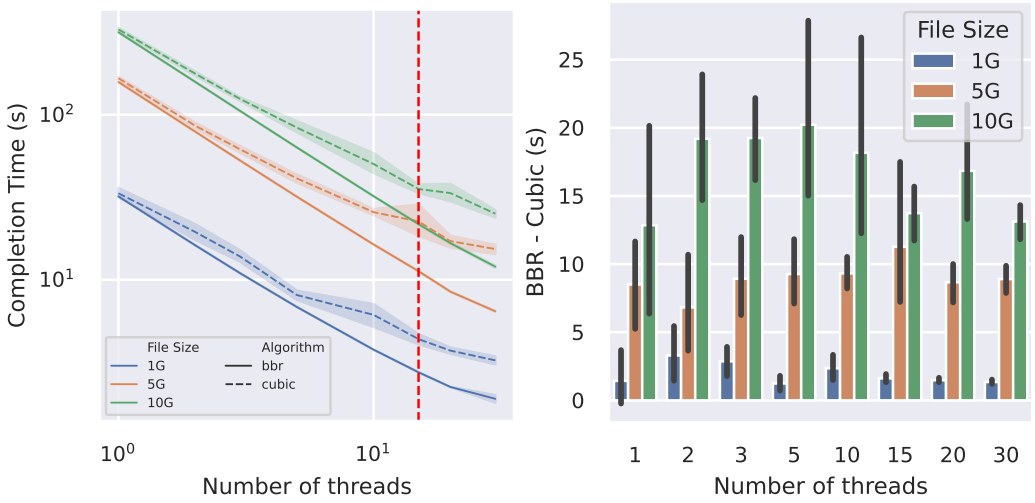

Figure 6. Flow completion time as function of multiple TCP streams (using threads) for Cubic vs. BBR (Left) and its differences (Right).

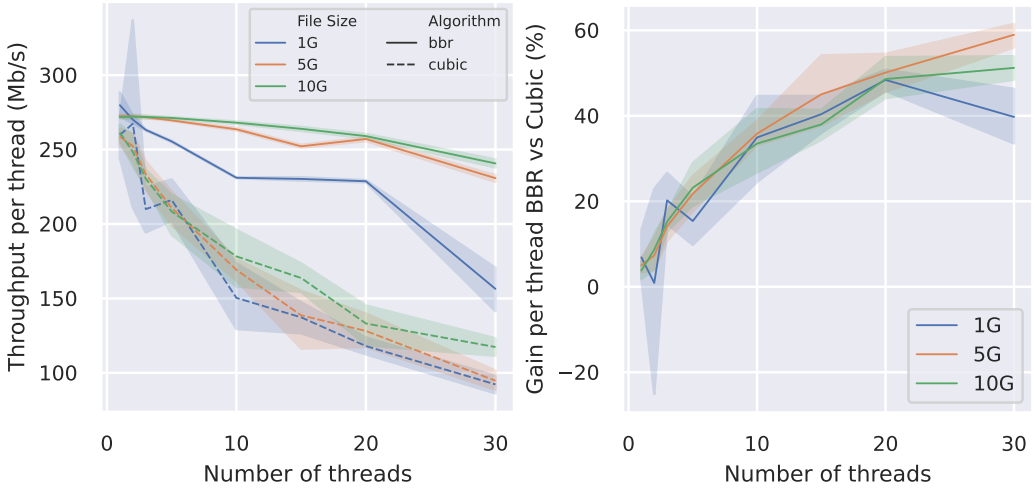

Figure 7. Throughput per thread (Left) and its gain for BBR vs Cubic (Right).

a second slice with a *multipath WAN topology* was created to perform a traffic engineer analysis, exploring the multiplicity of paths and the diversity of links between the hosts/DTNs.

The slice is a multipath topology composed by 7 nodes (`H1`, `R1`, `R2`, `R3`, `R4`, `R5`, and `H2`), respectively, geographically located in the following sites: `USCD` (USCD), `LOSA` (Los Angeles), `SEAT` (Seattle), `SALT` (Salt Lake City), `STAR` (StarLight), `DALL` (Dallas), and `MICH` (University of Michigan), as shown in **Fig. 8** (Left) and (Right). The Red, Green and Blue paths have respectively the average RTT of: 82.2 ms, 78.6 ms and 62.5 ms. We set the capacity of crossing links ($l1 = R3 \Longleftrightarrow R4$ and $l2 = R1 \Longleftrightarrow R5$) seeking for the point that achieves maximal performance for transferring the flows. We refer to an assignment of capacity values to the links as a slice configuration S_1. For example, in the configuration S_1, we set the links $l1 = R3 \Longleftrightarrow R4 = 50Mbps$ and $l2 = R1 \Longleftrightarrow R5 = 100$ from the topology of **Fig. 8** (Right), so that the ratio between the crossing links are $r = l1/l2 = 0.5$.

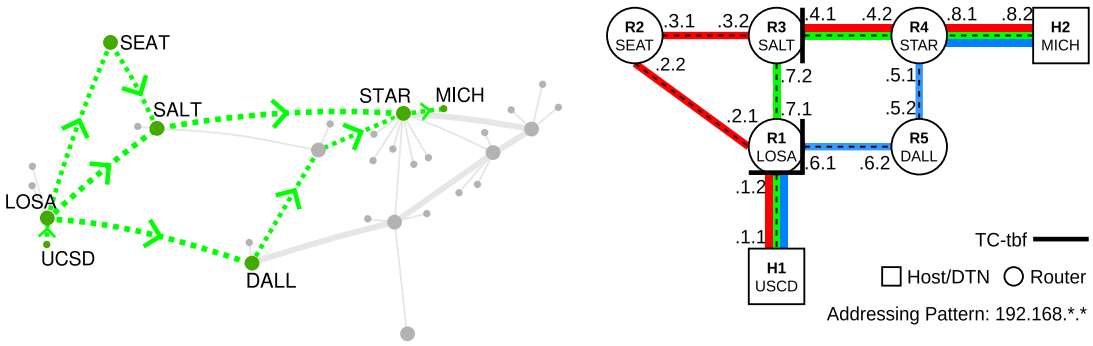

Figure 8. Multipath WAN deployment in FABRIC Testbed in (Left) with its logical slice varying links capacity in (Right).

The **Fig. 9** shows the result of sending 6 TCP cubic flows[6] of 1 Gigabyte each (as a typical DIS workload) by using multiple paths (more precisely two crossing links) between source and destination. The experiment is based on a classic load balancing algorithm, Equal-Cost Multi-Path (ECMP) [Hopps and Thaler 2000], which typically distributes traffic to all available paths in a round-robin fashion. This means that flow 1 is sent to the red path, flow 2 to the green path, and flow 3 to the blue path, and so forth. We tested various slice configurations within the range of $r \in \{0.5, 1 = uniform, \ldots, 2\}$.

As we can see in **Fig. 9**, the slice with a uniform bandwidth configuration ($r = 1$) does not necessarily yield the lowest flow completion time. In fact, the flow completion time decreases significantly as the ratio r increases, reaching the best performance when $r = 2$. For instance, the time to transfer the flows is the double for $r = 1$ (around 350s to complete all the flows) compared to $r = 2$ (around 175s).

This result is clearly explained by load balancing uniform flows over imbalanced paths ([Ros-Giralt et al. 2019]), where the ECMP algorithm splits the flows across the three paths, with two flows over $l1$ (red, green) and one flow on $l2$ (blue) per round. Nevertheless, it is crucial to find out the existence in practice of a wasteless point in bandwidth (known as proportional design [Ros-Giralt et al. 2021]) for a slice-specific configuration (for this case when $r = 2$). This proportional design allows us to make the best usage of the links *tuning a multipath slice*, which plays a crucial role in completing all the flows basically at the same time (e.g. for uniform flows[7]) without wasting of bandwidth.

5. Conclusion

Addressing the challenges related to the massive data transfer in DIS networks requires a holistic approach. While one might initially assume that existing networking techniques would seamlessly handle this task, the unique characteristics of data transport unveil complexities that demand specific considerations. Our exploration with a prototype slice, i.e., scaled down in terms of transport volumes present in DIS, already highlighted the need for effective bandwidth allocation management, especially for sets of interdependent transfers linked to scientific experiments. Additionally, the selection of an appropriate TCP

[6]We did also for BBR flows with slightly differences on performance.

[7]Non-uniform flows are intended for evaluation in future work.

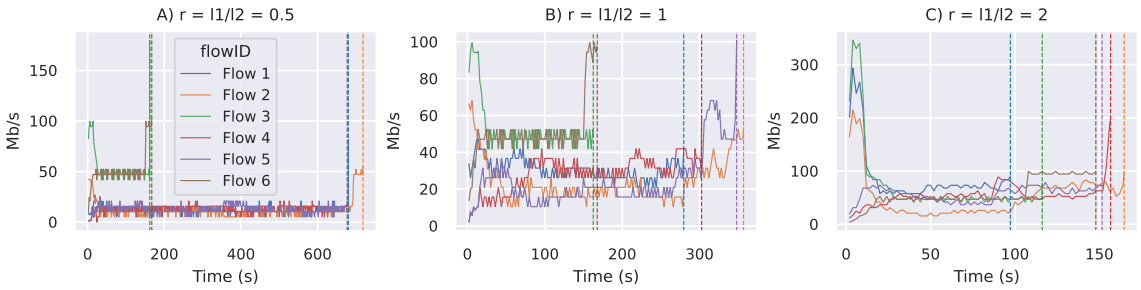

Figure 9. Flow completion time using ECMP algorithm for load balancing the flows in a WAN with different ratios of bandwidth: A) $r = l1/l2 = 0.5$**, B)** $r = l1/l2 = 1$ **and C)** $r = l1/l2 = 2$**).**

congestion control algorithm becomes crucial, particularly in scenarios involving a single path transfer.

To summarize the presented results, while slight differences favor BBR, the overall effectiveness of increasing the number of threads surpasses the impact of selecting TCP algorithms. However, in scenarios with a high number of threads and substantial data volume, BBR multiplexing appears to outperform Cubic. For multipath slice allocation, it is crucial to determine the practical existence of a bandwidth wasteless point, known as proportional design [Ros-Giralt et al. 2021], for a specific slice configuration. This design optimally utilizes links in tuning a multipath slice, playing a critical role in achieving nearly simultaneous completion of all flows, thus efficiently using available physical infrastructure bandwidth in DIS slices avoiding over-provisioning approaches.

As future works, we envision deploying PolKA source routing protocol [Dominicini et al. 2020] and its recent extension for multi-path [Guimarães et al. 2022], as a next step to advance our previous experience in protocol deployment at testbeds reported in [Borges et al. 2022]. We also plan to incorporate bottleneck structure analysis [Ros-Giralt et al. 2021] for the design of bandwidth-efficient slices and to optimize traffic engineering on a deployed slice. An example involves a distributed artificial intelligence (AI) workload, specifically the training of a large foundation model using globally-distributed computational nodes connected over a wide area network (WAN) [Yuan et al. 2023]. In such scenarios, efficient slices optimized for deterministic communication patterns like AllGather or AllReduce are essential. At last, we should evolve our prototype towards real DIS context, so we intend to use (Fast Data Transfer) FDT [Dunefsky et al. 2022] as a point of comparison. This tool can be adapted to incorporate strategies and different congestion control algorithms (e.g., BBRv3 [Cardwell et al. 2023]) or even the QUIC protocol [Iyengar and Thomson 2021].

6. Acknowledgments

This study received financial support from Brazilian agencies: CNPq, CAPES, FAPESP/MCTI/CGI.br (PORVIR-5G 20/05182-3, FAPES (94/2017, 281/2019, 515/2021, 284/2021, 06/2022, 1026/2022, 941/2022, #2022/NGKM5, #2021/GL60J) and SERPRO. Also, we express our gratitude to FABRIC for their support, with a special acknowledgment to Paul Ruth.

References

Baldin, I., Nikolich, A., Griffioen, J., Monga, I. I. S., Wang, K.-C., Lehman, T., and Ruth, P. (2019). Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing*, 23(6):38–47.

Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I. (2014). Geni: A federated testbed for innovative network experiments. *Computer Networks*, 61:5–23. Special issue on Future Internet Testbeds – Part I.

Borges, E., Pontes, E., Dominicini, C., Schwarz, M., Mate, C., Loui, F., Guimarães, R., Martinello, M., Villaça, R., and Ribeiro, M. (2022). A lifecycle experience of polka: From prototyping to deployment at géant lab with rare/freertr. In *Anais do XIII Workshop de Pesquisa Experimental da Internet do Futuro*, pages 35–40, Porto Alegre, RS, Brasil. SBC.

Both, C., Guimaraes, R., Slyne, F., Wickboldt, J., Martinello, M., Dominicini, C., Martins, R., Zhang, Y., Cardoso, D., Villaca, R., Ceravolo, I., Nejabati, R., Marquez-Barja, J., Ruffini, M., and DaSilva, L. (2019). Futebol control framework: Enabling experimentation in convergent optical, wireless, and cloud infrastructures. *IEEE Communications Magazine*, 57(10):56–62.

Cao, Y., Jain, A., Sharma, K., Balasubramanian, A., and Gandhi, A. (2019). When to use and when not to use bbr: An empirical analysis and evaluation study. In *Proceedings of the Internet Measurement Conference*, IMC '19, page 130–136, New York, NY, USA. Association for Computing Machinery.

Cardwell, N., Cheng, Y., Yang, K., Morley, D., Hassas, S., Jha, P., Seung, Y., Jacobson, V., Swett, I., Wu, B., et al. (2023). Bbrv3: Algorithm bug fixes and public internet deployment. *Presentation in CCWG at IETF*, 117.

Dominicini, C. et al. (2020). Polka: Polynomial key-based architecture for source routing in network fabrics. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 326–334. IEEE.

Dunefsky, J., Soleimani, M., Yang, R., Ros-Giralt, J., Lassnig, M., Monga, I., Wuerthwein, F. K., Zhang, J., Gao, K., and Yang, Y. R. (2022). Transport control networking: Optimizing efficiency and control of data transport for data-intensive networks. In *Proceedings of the ACM SIGCOMM Workshop on Network-Application Integration*, NAI '22, page 60–66, New York, NY, USA. Association for Computing Machinery.

Guimarães, R. S., Dominicini, C., Martínez, V. M. G., Xavier, B. M., Mafioletti, D. R., Locateli, A. C., Villaca, R., Martinello, M., and Ribeiro, M. R. N. (2022). M-polka: Multipath polynomial key-based source routing for reliable communications. *IEEE Transactions on Network and Service Management*, pages 1–1.

Hopps, C. and Thaler, D. (2000). Multipath Issues in Unicast and Multicast Next-Hop Selection. RFC 2991.

Huang, T., Yu, F. R., Zhang, C., Liu, J., Zhang, J., and Liu, Y. (2017). A survey on large-scale software defined networking SDN testbeds: Approaches and challenges. *IEEE Communications Surveys Tutorials*.

Iyengar, J. and Thomson, M. (2021). Rfc 9000: Quic: A udp-based multiplexed and secure transport. *Omtermet Emgomeeromg Task Force*.

M., S. et al. (2014). Design and implementation of the ofelia FP7 facility: The european openflow testbed. *Computer Network*, 61:132–150.

Martins, J. S. B., Carvalho, T. C., Moreira, R., Both, C. B., Donatti, A., Corrêa, J. H., Suruagy, J. A., Corrêa, S. L., Abelem, A. J. G., Ribeiro, M. R. N., Nogueira, J.-m. S., Magalhães, L. C. S., Wickboldt, J., Ferreto, T. C., Mello, R., Pasquini, R., Schwarz, M., Sampaio, L. N., Macedo, D. F., De Rezende, J. F., Cardoso, K. V., and De Oliveira Silva, F. (2023). Enhancing network slicing architectures with machine learning, security, sustainability and experimental networks integration. *IEEE Access*, 11:69144–69163.

Newman, H. B., Ellisman, M. H., and Orcutt, J. A. (2003). Data-intensive e-science frontier research. *Commun. ACM*, 46(11):68–77.

Ros-Giralt, J., Amsel, N., Yellamraju, S., Ezick, J., Lethin, R., Jiang, Y., Feng, A., Tassiulas, L., Wu, Z., Teh, M. Y., and Bergman, K. (2021). Designing data center networks using bottleneck structures. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 319–348, New York, NY, USA. Association for Computing Machinery.

Ros-Giralt, J., Bohara, A., Yellamraju, S., Langston, M. H., Lethin, R., Jiang, Y., Tassiulas, L., Li, J., Tan, Y., and Veeraraghavan, M. (2019). On the bottleneck structure of congestion-controlled networks. *Proc. ACM Meas. Anal. Comput. Syst.*, 3(3).

Salmito, T. et al. (2014). Fibre-an international testbed for future internet experimentation. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC 2014*, pages p–969.

Spang, B., Arslan, S., and McKeown, N. (2022). Updating the theory of buffer sizing. *SIGMETRICS Perform. Eval. Rev.*, 49(3):55–56.

Vargas, S., Drucker, R., Renganathan, A., Balasubramanian, A., and Gandhi, A. (2021). Bbr bufferbloat in dash video. In *Proceedings of the Web Conference 2021*, WWW '21, page 329–341, New York, NY, USA. Association for Computing Machinery.

Weigle, E. and chun Feng, W. (2002). A comparison of tcp automatic tuning techniques for distributed computing. In *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, pages 265–272.

Yuan, B., He, Y., Davis, J. Q., Zhang, T., Dao, T., Chen, B., Liang, P., Re, C., and Zhang, C. (2023). Decentralized training of foundation models in heterogeneous environments.

Zink, M., Irwin, D., Cecchet, E., Saplakoglu, H., Krieger, O., Herbordt, M., Daitzman, M., Desnoyers, P., Leeser, M., and Handagala, S. (2021). The open cloud testbed (oct): A platform for research into new cloud technologies. In *2021 IEEE 10th International Conference on Cloud Networking (CloudNet)*, pages 140–147.