Hercules: High-Speed Bulk-Transfer over SCION

Marten Gartner
OVGU Magdeburg

Jean-Pierre Smith
ETH Zürich

Matthias Frei
SCION Association

Francois Wirz
ETH Zürich

Cédric Neukom
ETH Zürich

David Hausheer
OVGU Magdeburg

Adrian Perrig
ETH Zürich

***Abstract*—With the steady increase in the resolution of cameras and screens, and the ever-expanding creation of information, data volumes have been rising exponentially. However, standard bulk-transfer still performs inefficiently and require complex configuration, especially for high-speed long-distance transfers. While next-generation Internet architectures promise significant improvements through path-aware networking (PAN), their adoption requires modern tools. To overcome these limitations, we propose Hercules, which combines efficient host networking and congestion control in the first high-speed bulk-transfer tool with native multipath support. We show that Hercules is outperforming bulk-transfer tools (GridFTP, bbcp) in intercontinental transfers, achieving up to 90% increase in goodput. Even in local testbeds, Hercules achieves competitive goodput in a 1500-byte MTU environment, and outperforms bbcp and GridFTP using a 3400-byte MTU. Finally, we demonstrate how Hercules natively aggregates multiple inter-domain paths while behaving fair to competing flows.**

***Index Terms*—High-speed networking, bulk transfer, path-aware networking, SCION Internet architecture**

I. Introduction

The topic of efficient utilisation of network resources has accompanied the Internet throughout its growth to over 4 billion users in the last 30 years. However, the last two decades witnessed a surge in the volume of generated data. Large corporations, such as Google and Amazon, built dedicated global networks to move their enormous data [1], [2]. Academic and research institutions deploy and interconnect networks to transfer petabytes of data necessary for modern scientific collaboration [3], [4]. Even operating a crypto-currency node may require downloading over 380 GB [5] at a time when shipping hard-disks is still common for transferring large amounts of data [6], [7].

While bulk transfer tools are designed to tackle these use-cases, in practice they suffer from a variety of limitations. At first, the ubiquitous TCP transport protocol is unable to efficiently utilise the network bandwidth, especially in high bandwidth-delay-product (*bdp*) networks, such as inter- or transcontinental networks. To overcome this issue, bulk transfer tools stripe their data across multiple TCP connections [8]–[10], which creates unfairness harming other TCP flows. Despite of the progress made in improving TCP on these networks [11]–[13], TCP remains inadequate for transferring bulk data as it still requires extensive tuning on end-hosts to achieve high performance [14]. Furthermore, UDP-based alternatives [15]–[17] remain constrained by the limits of general-purpose OS network stacks, not reaching sufficient performance for high-speed bulk transfer. Finally, path-aware networking (*PAN*) has been shown to improve transmission rates and reduce transmission times by bypassing network congestion [18], [19]. Simultaneous use of multiple paths accelerates transmission rates to fulfill performance requirements of modern bulk-transfer applications [20], [21]. Nevertheless, path selection and multipath have largely remained unsupported in the current Internet [22], [23].

To overcome these limitations, we design and implement Hercules, the first high-speed bulk transfer tool with native multipath support on top of a next-generation Internet architecture, incorporating state-of-the art components. Hercules defines a reliable, datagram-oriented protocol for bulk transfer that is layered upon UDP, and ensures fair use of shared network infrastructure by utilising Performance-oriented Congestion Control (*PCC*) [24]. To scale beyond the limitations of single-path architectures and to more efficiently utilise the available network bandwidth, Hercules forwards traffic across multiple paths using the path control provided by the SCION next-generation Internet. SCION allows inter-domain multipath without configuration changes in the network by incorporating the AS-level path into the packet header. Hercules leverages the recent Linux express data path (XDP) to achieve high packet processing performance – competitive with the hardware acceleration afforded TCP – while coexisting with the general OS network stack and without requiring exclusive access to the network interface. Finally, Hercules allows user-friendly file transfers by avoiding complex tuning on endhosts. To this end, we make the following contributions:

- We present the design, implementation and evaluation of Hercules, a high-speed bulk transfer tool based on the SCION next-generation Internet architecture.
- We show that Hercules outperforms existing bulk transfer tools [9], [25], while incorporating the benefits of path-aware networking.

The remainder of this work is structured as follows: In Section II, we provide required background focusing on SCION, PCC and AF_XDP. After discussing limitations of modern bulk transfer tools in Section III, we present our Hercules design in Section IV. Afterwards, we evaluate Hercules against state-of-the-art bulk transfer tools in Section V, followed by

ISBN 978-3-903176-57-7

related work in Section VI and conclusions in Section VII.

II. Background

The SCION Internet architecture [26], PCC [24], and Linux's express data path [27] form the foundation for Hercules.

A. The SCION Internet Architecture

SCION [26] is a clean-slate Internet architecture designed to provide route control, failure isolation, and explicit trust information for end-to-end communication. SCION enables path-aware networking through implementing *packet carried forwarding state*, where every packet contains a compact representation of the complete inter-domain path to the intended destination (in the form of *hop fields*) in its packet header. Each hop field represents an AS on the path. Through cryptographic mechanisms, the SCION architecture ensures that a packet can not depart from its specified path along the way. Consequently, SCION applications can ensure that packets traverse the Internet over the selected path. Distributing traffic over multiple paths can consequently be implemented in SCION by directing packets via different paths, i.e., by encoding the particular paths into the SCION header when creating the packets. SCION hosts can choose to use one or more paths to each destination, using different paths for different purposes or even multiple paths for a single purpose. Moreover, the sender is aware of path failures or new availabilities and can thus react appropriately, for example, by switching paths. SCION is currently deployed in a global production network [28] as well as a research testbed [29].

B. Performance-oriented Congestion Control (PCC)

PCC [24] is a congestion control algorithm based on empirical observations of the performance resulting from changes in the sending rate. It evaluates the benefit of increasing or decreasing the rate using a series of paired trials. For a transmission rate r and $\epsilon < 1$, it sends at the rate $r \cdot (1 + \epsilon)$ for an interval, and at $r \cdot (1 - \epsilon)$ for another interval. Then, for each trial, PCC collates the acknowledgements and assesses the utility (a function of throughput and loss) of the action in that interval. After multiple such pairs of trials, PCC selects the action (increasing or decreasing) that resulted in greater utility. After selecting an action, PCC begins adjusting its sending rate in that direction. If the utility decreases, PCC starts again with a new series of trials.

By basing decisions on these trials, PCC ignores spurious losses and better utilises the available bandwidth across various network environments.

C. Linux's Express Data Path (XDP)

XDP [27] enables high-performance networking by circumventing the traditional OS network stack and providing a direct interface to the network device for user space programs. To avoid data copies, an XDP socket shares a memory buffer, the UMEM, between the user-space program, the kernel, and the network device. The UMEM is divided into equally sized frames that can be used for receiving or transmitting data and utilises ring buffers for coordination between the program, kernel, and network device. By bypassing the network stack, XDP allows a program to achieve transfer rates close to the line rate of the network device, but without taking ownership from the kernel and preventing other programs from using the device.

III. Limitations of Long Distance High-Speed Bulk Transfer

We observe four core limitations of long distance high-speed bulk transfer, which we discuss in this section:

1) TCP in high bandwidth-delay product (bdp) environments: TCP usually reacts to particular network events, e.g., packet loss or RTT increase, by changing the size of the congestion window. However, this window-based approach shows problems filling high bdp links [30], in particular reducing the window size in case of random packet loss. Consequently, bulk transfer tools often run a large number of parallel flows, behaving unfairly to competing flows.

2) TCP needs high tuning effort: To saturate modern 40+ Gbps network links, TCP requires specific tuning on endhosts. Particular parameters like buffer sizes need to be configured on each host and congestion control algorithms need to be set according to current network settings. Finally, also bulk transfer applications require configuration to perform best e.g., window size for bbcp. While applying proper tuning increases performance in local networks, modern bulk transfer tools still suffer from TCP's poor performance on high bdp links.

3) UDP limited by general-purpose network stacks: UDP-based transport protocols offer high flexibility by avoiding the strict congestion control model of TCP, showing significant improvements in, e.g., global Internet traffic [31]. However, UDP-based approaches are missing the performance improvements that TCP stacks experienced and are often limited by general purpose OS network stacks [16], [17].

4) Multipath capabilities: While emerging approaches (e.g., mpath [19]) provide application-layer approaches, these are not deployed widely or cannot provide sufficient performance. MPTCP [32] and MPQUIC [33] enable multipath at interface-level, while MPTCP requires a dedicated network interface per path. Both approaches define a path as an outgoing interface and are consequently unable to achieve multipath over a single interface. To enable end-to-end multipath, intricate configuration in the network is required. Finally, MPTCP implements a pessimistic congestion control in the sense that multiple MPTCP flows can achieve at most as much bandwidth as a single TCP flow at shared bottlenecks, which can erroneously slow down the sending rate in some use-cases.

IV. Hercules

Hercules is a high-volume data-transfer tool with integrated multipath capabilities and congestion control, enabling efficient bulk transfer in high-speed local and inter-domain

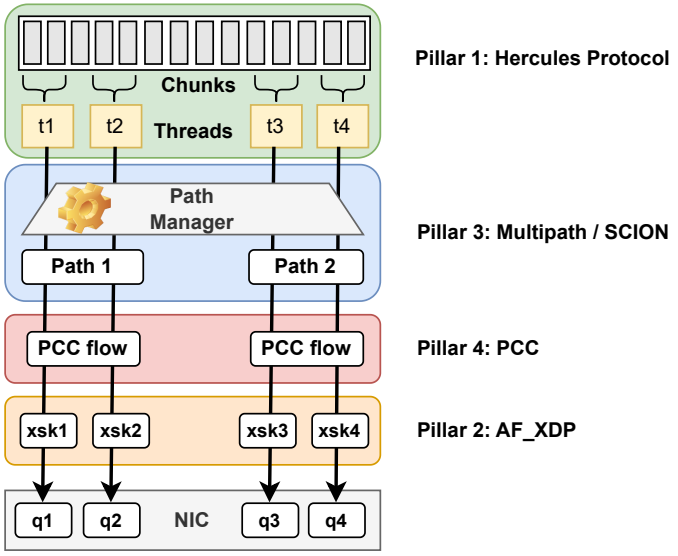

Fig. 1: Overview of the Hercules architecture

networks. Hercules consists of four core components, which we refer to as the four pillars:

1) Hercules Protocol: Our custom Hercules protocol is a performance-oriented bulk transfer protocol based on Reliable Blast UDP [15], that transfers chunks of a file over a series of rounds.

2) High Performance through AF_XDP: To achieve high performance network throughput, Hercules employs Linux's express data path (XDP) [27] socket type to bypass the traditional in-kernel network stack and achieve high transfer rates from user space.

3) Multipath via SCION: Hercules can utilise multiple network paths and provides a simple interface for specifying path preferences. Thus, it enables routing traffic around bottlenecks or distrusted parties by being built on top of the SCION network architecture.

4) PCC Congestion Control: Hercules is deployable in public networks as it utilises the PCC [24] algorithm to distribute network resources across competing flows.

Figure 1 shows how the four pillars of Hercules interact. In the sending direction, the protocol splits the file into chunks. Multiple worker threads handle the chunks and interact with the path-manager to schedule chunks over the selected paths. The path manager allocates memory for a packet for the respective chunk and adds the proper path headers. This memory block is handed over to the following layers, starting with the PCC layer. Hercules creates a dedicated PCC flow for each path. Finally, packets traverse Hercules' AF_XDP layer, where each thread owns one XDP socket (xsk), which is responsible for writing packets to the NIC. On the receiving side, data flows in the opposite direction: Packets are read from the NIC by one or more xsks and traverse PCC congestion control. For each packet, the receiving thread copies the chunk

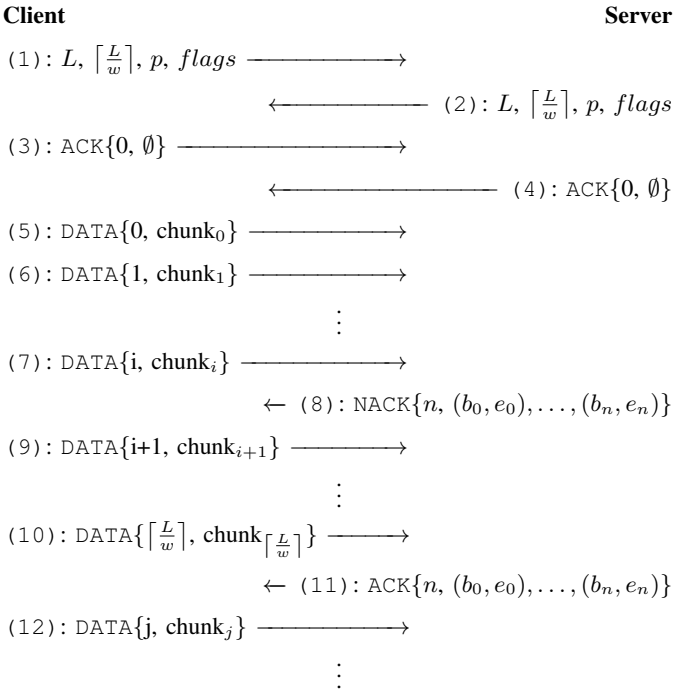

Fig. 2: Example Hercules protocol trace showing connection establishment and transfer of a file of size L bytes and payload (chunk) size of w bytes.

from the packet into the proper slot of the destination file. The layers again interact through passing the memory location of the packet.

Hercules also supports transferring a file to multiple destinations in parallel, while supporting multipath communication to each destination. The four pillars are instantiated for each destination (while keeping a single instance of the file in main memory) and operate independently of each other.

A. The Protocol

Hercules operates in a client-server model, where the client acts as the sender, transmitting a file to the server that acts as the receiver. Both are engaged for a single transmission.

1) Overview: Hercules transmits over a series of rounds. After a handshake, the first round begins and the sender sequentially transmits the file's chunks, intermittently adjusting the sending rate based on the feedback of the congestion control. Meanwhile, the receiver periodically acknowledges chunks. After the sender has transmitted all the chunks, a new round begins where chunks lost in the previous round are retransmitted. This procedure repeats until the entire file has been acknowledged by the receiver, at which point, both endpoints terminate. This round-based approach, inspired by the constant-rate bulk-transfer protocol RBUDP [15], reduces the number of control messages on the network in delay-insensitive applications such as bulk-transfers.

We next describe the connection and path establishments, congestion control, and reliability mechanisms.

2) Connection Establishment: A Hercules connection is established over UDP using four messages exchanged between the client and server, as well as the reservation of resources

for the connection, as depicted by Messages 1–4 of Figure 2. The connection is initiated by the client for the transmission of a file of size L. First, the client divides the file into $\lceil \frac{L}{w} \rceil$ chunks, each at most w bytes in length. The MTU and the path header size determine the possible chunk size. Next, the client notifies the server of L and w using a handshake packet (m.1), which the server immediately echoes (m.2) and proceeds to prepare a file mapping for storing the received file. The immediate echo allows the client to calculate rtt_{C}, an initial estimation of the round-trip-time (RTT) for initialising transport parameters. Similarly, the client sends an empty acknowledgement on receipt of the echoed packet (m.3) to allow the server to calculate its RTT estimation rtt_{S}, which is required by congestion control (see Section IV-D). Once the server completes the space allocation for the file, it sends an empty acknowledgement to signal that it is ready to receive data (m.4).

3) Data Transmission and Reliability: Data packets in Hercules contain the index of the chunk being transmitted along with the data. In contrast to TCP, Hercules is not a windowed protocol. Although the sender transmits chunks sequentially, the receiver is always prepared to receive any chunk. On receipt of a chunk, the receiver immediately writes the chunk to its location in the file, identified by the chunk index.

Hercules transmits data in a series of rounds. In the first round, the sender transmits the entire file, without retransmission (`DATA` Messages 5–10). In the second round, the sender retransmits lost chunks after aggregating the acknowledgements from the receiver (Messages 12 onward). Unlike in RBUDP, the receiver does not wait until the end of the round to acknowledge data. Instead, the receiver periodically acknowledges chunks using acknowledgement packets (m.11), which confirm the receipt of ranges of chunk-ids. Additionally, the receiver sends negative acknowledgements (NACKs) to signal lost packets whenever packets arrive out-of-order on a path to the receiver. Subsequent rounds are comprised of the retransmission from the previous round. With these adaptations to RBUDP, we obtain the properties needed by Hercules' strict performance requirements, whereas UDT [17] employs window-based flow control and SABUL [16] uses TCP for control messaging.

B. High Performance through AF_XDP

Recent works show that AF_XDP as kernel fast path allows significantly higher networking speeds than regular posix sockets [27]. However, we observe that for our complex use case of a reliable, path-aware bulk transfer protocol, system design and application tuning are essential. We integrate AF_XDP into Hercules with close attention to efficiently implementing main-memory access through the whole sending and receiving process. Especially queue selection, multi-threading and cache alignment play an important role for Hercules tuning. Consequently, we perform the following optimization techniques to send and receive at high rates:

a) SCION header caching and optimized SCION stack: Hercules pre-computes the SCION header for each path (which does not change over the lifetime of the connection) and re-uses the cached header for each packet on a particular path. Furthermore, the SCION dispatcher limits the overall endhost performance [34] and needs to be bypassed for high speeds.

b) Efficient multithreading: Packet creation on the client and packet parsing on the server side is distributed over multiple threads. Each thread manages its own XDP socket. The usage of multiple XDP sockets provides important optimization potential: On the client side, multiple cores can be used in parallel (each socket is bound to a particular core) to increase throughput. On the server side, receive-side scaling (RSS) allows to distribute packets to XDP sockets while efficiently distributing load over multiple cores.

c) Cache-aligned sending queues: To efficiently support multiple worker threads, we implement cache-aligned, thread-safe send queues, which optimize the interaction of multiple threads with XDP by reducing cache misses.

C. Multipath via SCION

As presented in Section II, a SCION path consists of multiple hop fields, each representing an AS on inter-domain path, encoded in an efficient way into the SCION Header [26] of each packet. This allows SCION applications to direct a SCION packet via a particular path by encoding the respective hop fields into the SCION header, giving full control over the inter-domain paths to applications.

Hercules uses SCION's path-awareness primarily to aggregate the capacities of multiple paths. In addition to the complete path encoded in the SCION Header, Hercules adds the path index field p to the packets to simplify the identification of paths (i.e., to reduce the overhead of using the complete path as identifier).

At the beginning of the transfer, Hercules performs a separate handshake on each path to probe its connectivity and calculate its RTT. After accepting the handshake on a particular path, the server sends its control messages over the path on which the data was received. Given the diverse RTTs and bandwidths of paths, each path is congestion-controlled independently, the congestion control algorithm is initialised using the RTT inferred from the path handshake. After the client receives a handshake response, the respective path is enabled and the data transfer over this path starts.

The use of independent congestion-control algorithms on each path results in Hercules receiving bandwidth proportional to the number of paths traversing a shared network bottleneck. While this may be unfair to single-path protocols, most data transfer tools similarly employ TCP by striping data across independent connections [8]–[10]. Using SCION's path information, however, Hercules could detect and avoid shared or congested inter-domain links. Each hop on the SCION path contains the ingress and egress interface numbers of the respective AS, which create a unique identifier combined with the AS number. In combination with monitoring the enabled

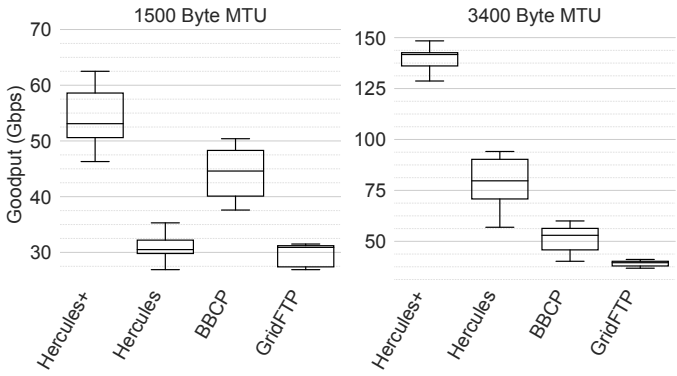

Fig. 3: High-speed LAN M2M goodput for 1500- and 3400-byte MTU sizes (200 Gb*p*s link, <1 ms RTT), 4 parallel flows.

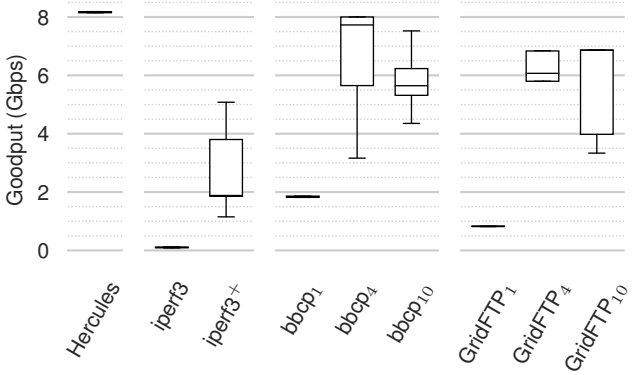

Fig. 4: Goodput from Hong Kong to Chicago (10 Gb*p*s, 194 ms RTT). Subscripts denote parallel TCP flows.

paths, Hercules could be able to detect shared bottlenecks through these identifiers, i.e., by intersecting the identifier list of multiple paths.

D. Congestion Control

Hercules uses PCC [24] to regulate its sending rate in response to network congestion. PCCs core idea is to adjust the performance of a flow by results obtained from a set of trials with varying sending rate. Rate adjustments in PCC occur periodically as governed by the measurement interval I. PCC performs trials over multiple measurement intervals to determine whether to change the current sending rate, and in which direction the rate needs to be changed. In Hercules, each path is congestion-controlled independently. We set I to a multiple (a random factor in the interval $[1.7, 2.2]$ [24]) of the initial round-trip time rtt_C, obtained from each path's handshake. Hercules aggregates the acknowledged chunks as well as the NACKs to calculate the utility of each current sending rate. While the NACKs are sent back to the client immediately to notify about packet loss, Hercules bulk transfer nature allows to delay acknowledgements to be sent at the end of the measurement intervals, since the transfer size is known on the server side. This provides synergies with PCC, since acknowledgements are only required at the end of each interval. Furthermore, Hercules sends significantly fewer acknowledgements, which reduces the load on the reverse link.

V. Evaluation

In this section we evaluate Hercules' performance compared to state-of-the-art bulk transfer tools GridFTP [25] and bbcp [9], as well as iperf3 [35] in a set of experiments. We run iperf3 with default system settings and also with tuned TCP settings called *iperf3*$^+$ following the guidelines set forth by the Energy Sciences Network (ESNET) [14]. GridFTP and bbcp also use tuned TCP settings.

We present three core types of experiments to evaluate Hercules' performance. We start with an intra-lab high-performance experiment, followed by two experiments running large distance transfers. Finally, we perform multipath transfers over SCION, evaluating Hercules' multipathing capabilities and its fairness to competing flows.

A. High-Performance Intra-Lab Transfers

Researchers and institutions often need to transfer large quantities of data between different hosts within the same institution. We therefore ascertain Hercules' single-path limits in a high-speed intra-lab setup. Despite Hercules being optimized for high-bdp links, we aim to achieve at least comparable performance to GridFTP and bbcp in local networks.

We evaluate Hercules between two adjacent servers with 1500 and 3400 Ethernet frame sizes (MTUs).[1] Each server is equipped with AMD EPYC 7543P 32-Core processors (up to 3.70 GHz), 128 GiB of RAM (8 banks), and 200 Gb*p*s Mellanox ConnectX-6 cards, which are directly connected via a 200 Gb*p*s Ethernet link with an average round-trip-time of 245 µs. To better utilise the CPU, we bind the transfer tools to the same CPUs as the network interface card using the numactl command line utility (except GridFTP which does not support easily changing the binary to be executed).

Figure 3 shows the goodput of the transfer tools with 4 parallel flows at 1500-byte and 3400-byte MTUs. We run Hercules in a configuration without PCC (Hercules+) and with PCC. With a 1500-byte MTU, Hercules and GridFTP achieve a comparable median goodput of 31 Gb*p*s, bbcp 45 Gb*p*s and Hercules+ 52 Gb*p*s. With a 3400-byte MTU, Hercules+ clearly outperformed the other tools achieving 140 Gb*p*s median goodput, while Hercules achieves 80 Gb*p*s. Bbcp and GridFTP result in slightly higher performance compared to the 1500-byte MTU. These results prove that Hercules is in these lower-BDP environments comparable to the existing TCP network stack, and for larger MTUs outperforms other TCP-based systems.

B. Intercontinental Transfers

International research collaborations often co-fund and operate shared infrastructure such as the ITER project [37] and CERN. The massive amounts of data generated by these infrastructures must often be transferred thousands of kilometres

[1] XDP currently only supports up to around 3400 byte frames [36].

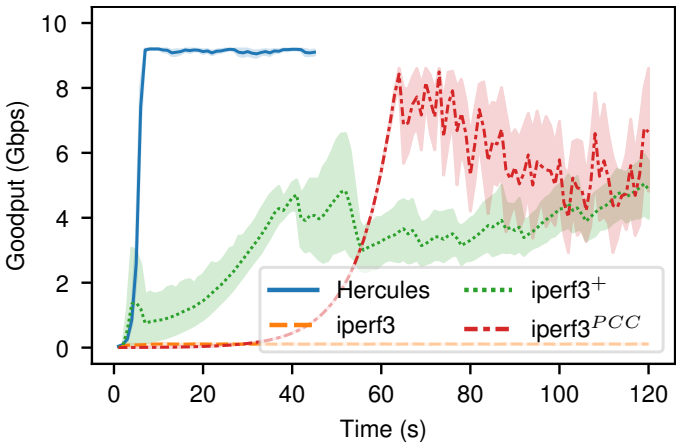

Fig. 5: Intercontinental M2M goodput over time from Hong Kong to Chicago (10 Gb*ps*, 194 ms RTT).

over research networks. We therefore evaluated Hercules on the South Korean academic network KREONet2 [38], which interconnects research networks globally. For each candidate, we performed 10 transfers eastward from Hong Kong to Chicago, United States via Daejeon, South Korea; a distance of over 12 500 km with an RTT of 194 ms. Each endpoint was equipped with an Intel Xeon Silver 4114 CPU (2.20 GHz), 16 GiB of memory (1 bank), and an Intel X710 10 Gb*ps* Ethernet card.

At 10 Gb*ps* and almost 200 ms RTT, KREONet2 has a bandwidth-delay-product of around 242.5 MB. Figure 4 shows the goodput in this setting. Losses resulted in median transfer rates below 2 Gb*ps* for the single TCP flows (iperf3, bbcp, and GridFTP), thus requiring the use of multiple TCP flows ($\text{bbcp}_{4,10}$, $\text{GridFTP}_{4,10}$). However, we observe that the performance stagnates for more than 4 flows for bbcp and GridFTP, due to many flows canceling potential performance gains through congestion. In contrast, Hercules maintained above 8 Gb*ps* with a single flow, thereby outperforming multiple competing TCP flows.

The change in goodput over time in Figure 5 provides further insights about the performance of Hercules and iperf3. We increase the runtime for iperf3 to 120 s and incorporated iperf3 with PCC. We observe that iperf3 with H-TCP [13] required over 40 s to increase its transmission rate to around 5 Gb*ps*. While iperf3 with PCC performed better, it still required over a minute to exceed 8 Gb*ps* before slowly decreasing towards 6 Gb*ps*. By contrast, within 10 s Hercules exceeded ∼9 Gb*ps* and remained above it for the rest of the transmission. Hercules performs better than iperf3 with PCC, since Hercules combines the userspace implementation of PCC with high-speed and low-latency networking. The PCC kernel module implementation used in iperf3 in contrast was significantly adapted [39] due to missing features in kernel modules, i.e., per packet state and proper RTT estimations.

To further analyze the performance at higher network speeds, we perform 10 transfers for each candidate between two machines on a 100 Gb*ps* link over the Atlantic. Figure 6 depicts the results, all candidates use 4 parallel flows. Each

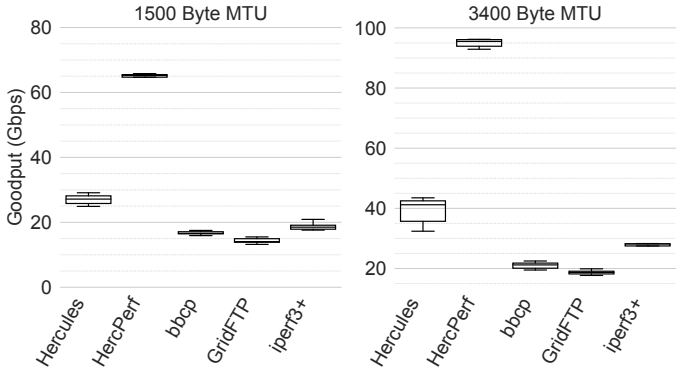

Fig. 6: Goodput of large-distance transfer over the Atlantic (100 Gb*ps*, 152 ms RTT).

server is equipped with an Intel(R) Xeon(R) Gold 6230 processor (20 cores, up to 3.90 GHz), 32 GiB of RAM (1 bank), and 100 Gb*ps* Mellanox ConnectX-5 interface cards. We observe that Hercules outperforms the other candidates achieving 27 Gb*ps* median goodput for 1500-byte MTU. GridFTP results in around 14 Gb*ps* median goodput and bbcp in 16.5 Gb*ps* for 1500-byte MTU. With a 3400-byte MTU, Hercules achieves 41 Gb*ps* median goodput, compared to 22 Gb*ps* for bbcp and 18 Gb*ps* for GridFTP, resulting in a performance increase of around 90% and 125%, respectively. Since we suspect that the available main-memory bandwidth of a single memory bank is limiting Hercules' performance, we run the same experiment with iperf3$^+$ and *HercPerf*, a variant of Hercules that sends dummy packets instead of actual payload, avoiding the copy of chunks on both sides. However, the remaining architecture of Hercules stays unchanged, which allows a fair comparison against iperf3$^+$. Our assumption about the memory bottleneck is confirmed as the avoidance of in-memory copies of chunks allows Hercules to achieve around 95 Gb*ps* goodput, while iperf3$^+$ only marginally outperforms bbcp and GridFTP. We observe, that without the limitation of a single memory bank, Hercules is able to fill the available 100 Gb*ps* link.

C. Multipath Transfer over SCION

After analyzing the short and long distance, single-path experiments, we evaluate Hercules' native multipath capabilities measuring its performance across 20 transfers utilising two SCION paths over the transcontinental academic network GÉANT. Figure 7 depicts the network topology used in this experiment.

The transfer starts in Geneva, CH over a single 10 Gb*ps* link to the next border router in CERN, which provides two SCION paths to Amsterdam, NL. One path traverses Paris, FR and the other path Hamburg, DE. Both paths provide similar RTTs of 22.9 ms and 24.5 ms, respectively. Since each location was provisioned with a single 10 Gb*ps* duplex link, the packets leaving Geneva on both the Paris and Hamburg paths share Geneva's outgoing 10 Gb*ps* capacity. While running Hercules over two fully disjoint paths with 10 Gb*ps* will likely double

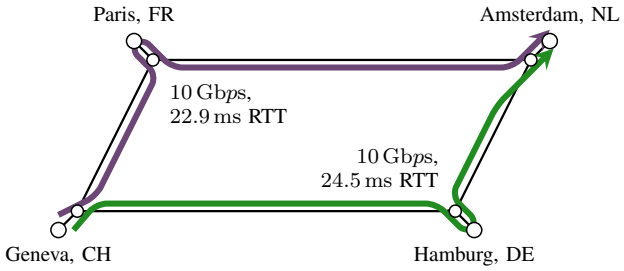

Fig. 7: Topology of the utilised GÉANT nodes. Data is transferred from Geneva, CH to Amsterdam, NL over both available paths.

the throughput, we decide to include a shared bottleneck to evaluate how Hercules combining multiple PCC flows behave in this scenario. Furthermore, 15 s after the transmission's start, we reduced the available bandwidth of one path by congesting the link at Hamburg for 15 s with UDP traffic. We anticipate that Hercules is capable of reacting fairly to the congestion by shifting the majority of its traffic to the path over Paris.

For non-SCION IP traffic there is only a single path available from Geneva to Amsterdam (over Paris). Comparing a TCP-based system against Hercules in this setting will result in unfair outcomes, since singlepath traffic can either run completely over the congested path or over the uncongested one. While the first scenario gives Hercules an unfair advantage, the latter would not include any congestion for the TCP-based system. Consequently, we decide to also configure the two paths for non-SCION traffic and compare Hercules against two independent iperf3$^+$ flows, one for each path. In contrast to untuned iperf3, we expect iperf3$^+$ to completely saturate the available bandwidth, to create a fair comparison against Hercules. While Hercules is capable of aggregate both SCION paths without any configuration effort in the network, by simply specifying the usage of both paths, we were required to apply additional configuration to the intermediate nodes to create two end-to-end paths via dedicated VLANs.

Figure 8 shows the per-path and aggregated throughputs achieved by Hercules and the two independent, tuned TCP flows of iperf3$^+$. Both tools increase their throughput to around 5 Gbps per path in the first seconds, fully saturing the available 10 Gbps. When faced with congestion on one path starting at 15 s, Hercules and iperf3$^+$ increased transmission on the second path to fully use the available capacity of Geneva's outgoing link. Finally, after the congestion ends, Hercules and iperf3$^+$ adjust their throughput back to 5 Gbps per path. The slower reaction time of Hercules is likely due to PCC, which is known to react more slowly to changes in network conditions [40].

Through Hercules' utilization of both paths, it was able to maintain a total minimum throughput of 7.5 Gbps during the congested period, and an overall throughput of 8.6 Gbps. We show, that Hercules is natively able to aggregate the bandwidth of both paths (despite being limited by the shared bottleneck in Geneva) and react fairly to congestion through a failover to

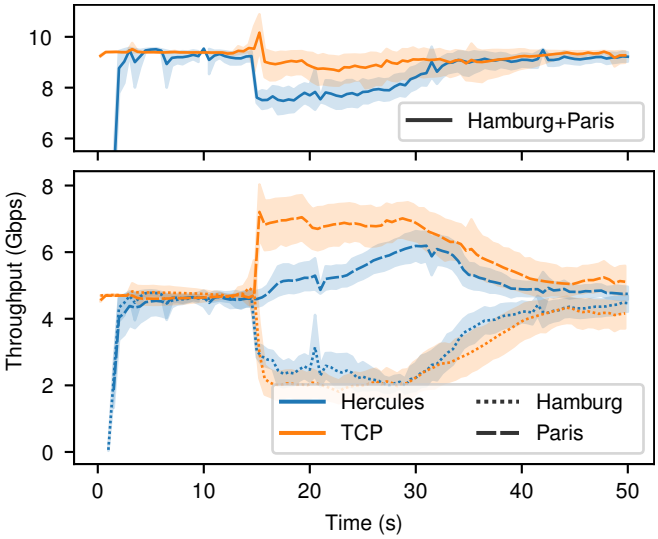

Fig. 8: Per-path and total throughput of Hercules and two independent TCP flows running over GÉANT.

the uncongested path. Although the two tuned iperf3$^+$ flows provide similar results, these are only possible in the TCP/IP network with manual network configuration.

From this experiment, we conclude that Hercules native multipath provides the anticipated results by simply configuring it to use both paths in parallel. Iperf3$^+$ was able to make use of both available paths through manual configuration in the network. While this is possible in smaller networks, the configuration complexity is increasing significantly in larger networks. Finally, we expect promising benefits from Hercules native multipath support on top of the SCION Internet architecture in Internet-scale networks, allowing bandwidth aggregation over larger numbers of heterogeneous paths and shared bottleneck detection based on the contained hops in the SCION header.

VI. Related Work

Works attempting to satisfy the need for high-BDP bulk transfer consists of single-flow and multiple-flow approaches.

a) Single-Flow Approaches: Single-flow approaches refine TCP or provide an alternate protocol. H-TCP [13], HSTCP [12], Scalable TCP [41] and CUBIC [11] improve TCP's throughput in high-BDP networks by being more aggressive for large congestion windows, thereby quickly filling available bandwidth; FAST TCP [42] and BBR [43] use delay to detect network congestion; and the PCC algorithms [24], [40] employ online-learning to find ideal sending rates.

Unfortunately, approaches using single TCP connections require extensive network-stack tuning and do not scale to high-BDP links [44], [45]. Reliable Blast UDP [15] sends UDP packets at a pre-configured rate and performs retransmissions at the end of the file transfer. SABUL [16] adds congestion control and periodic acknowledgements to RBUDP for use on shared links, and UDT [17] further improves congestion avoidance.

Although Hercules is a new UDP-based protocol, it leverages advancements in congestion control to enable use on shared links, provides a more performant user-space implementation, and can use multiple network paths.

b) Multiple-Flow Approaches: Other schemes employ multiple flows either on a single path through the network, known as striping, or on multiple network paths. TCP striping [46] is more common and is employed in data-transfer tools such as GridFTP [8], bbcp [9], mdtmFTP [10]. In TCP striping, the sending program opens multiple TCP connections, divides the file to be transferred into parts, and transmits them in parallel over the connections. Unfortunately, TCP striping improves performance at the expense of other TCP flows, dominating shared network bottlenecks.

Multipath TCP [32] defines a path as the interface over which data is transferred. For mTCP [18] and mPath [19], which utilise one-hop in-network overlays to provide divergent IP routes. Finally, other approaches, such as Huang *et al.* [23] and Phoebus [22], integrate with the network to control the network path. Hercules allows both simultaneous use of multiple network paths and congestion-controlled transmission in public networks.

Finally, gfal2 [47] integrates multiple backends into a single library, allowing applications to easily switch between protocols.

VII. Conclusions

With the expanding deployment of path-aware networks (PAN), exciting opportunities arise for higher performance and higher quality of communication. In this paper, we present Hercules, a new bulk transfer system that leverages a combination of cutting-edge networking technologies of the past decade, including kernel bypass high-speed networking, performance-oriented congestion control (PCC), and path-aware networking. The intricate engineering behind Hercules outperforms established systems such as GridFTP and BBCP by a staggering 90% in a large-distance environment. Hercules allows inter-domain path aggregation and failover without configuration changes in the network, through a simple configuration interface.

The Hercules code is available open-source [48]. In future work, we plan to investigate what other networking applications can benefit from these new technologies.

References

[1] S. Jain, A. Kumar, S. Mandal, *et al.*, "B4: Experience with a globally-deployed software defined WAN," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013.

[2] Amazon Web Services. "AWS global network." (2022), [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/global_network/.

[3] CERN, *Cern monitoring*. [Online]. Available: https://monit.web.cern.ch/public-site/img/gallery-5.png.

[4] GÉANT Association. "GÉANT network." (2022), [Online]. Available: https://network.geant.org/.

[5] Blockchain.com. "Blockchain size (MB)." (2022), [Online]. Available: https://www.blockchain.com/charts/blocks-size.

[6] Microsoft, *What is Azure import/export service?* 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/storage/common/storage-import-export-service.

[7] Amazon Web Services, *AWS Snowcone*, 2020. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2020/06/aws-announces-aws-snowcone/.

[8] B. Allcock, J. Bester, J. Bresnahan, *et al.*, "Data management and transfer in high-performance computational grid environments," *Parallel Computing*, vol. 28, no. 5, pp. 749–771, 2002.

[9] Stanford University, *bbcp*, version 15.02.03.00.1, Feb. 2015. [Online]. Available: https://www.slac.stanford.edu/~abh/bbcp/.

[10] L. Zhang, W. Wu, P. DeMar, and E. Pouyoul, "mdtmFTP and its evaluation on ESNET SDN testbed," *Future Generation Computer Systems*, vol. 79, pp. 199–204, 2018.

[11] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[12] S. Floyd, "Highspeed TCP for large congestion windows," IETF, RFC 3649, Dec. 2003.

[13] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proc. PFLDnet, Argonne, 2004.*, 2004.

[14] The Energy Sciences Network (ESnet). "Host tuning - Linux tuning." (Sep. 3, 2021), [Online]. Available: https://fasterdata.es.net/host-tuning/linux/.

[15] E. He, J. Leigh, O. Yu, and T. A. Defanti, "Reliable Blast UDP: Predictable high performance bulk data transfer," in *Proceedings. IEEE International Conference on Cluster Computing*, Sep. 2002, pp. 317–324.

[16] Y. Gu and R. Grossman, "SABUL: A transport protocol for grid computing," *Journal of Grid Computing*, vol. 1, no. 4, pp. 377–386, 2003.

[17] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks*, vol. 51, no. 7, pp. 1777–1799, 2007.

[18] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *2004 USENIX Annual Technical Conf. (USENIX ATC 04)*, ser. ATEC '04, Boston, MA: USENIX Association, 2004.

[19] Y. Xu, B. Leong, D. Seah, and A. Razeen, "mPath: High-bandwidth data transfers with massively multipath source routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 10, pp. 2046–2059, Oct. 2013.

[20] K. Yamanaka, H. Nakanishi, T. Ozeki, *et al.*, "High-performance data transfer for full data replication between ITER and the remote experimentation centre,"

Fusion Engineering and Design, vol. 138, pp. 202–209, 2019.

[21] National Institute of Informatics. "MMCFTP file-transfer protocol achieves transmission speeds of 231 Gbps, A new world record for long-distance data transmission." (Dec. 2017), [Online]. Available: https://www.nii.ac.jp/en/news/release/2017/1214.html.

[22] E. Kissel, M. Swany, and A. Brown, "Phoebus: A system for high throughput data movement," *Journal of Parallel and Distributed Computing*, vol. 71, no. 2, pp. 266–279, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731510001723.

[23] C. Huang, C. Nakasan, K. Ichikawa, and H. Iida, "A multipath controller for accelerating GridFTP transfer over SDN," in *2015 IEEE 11th International Conference on e-Science*, Aug. 2015, pp. 439–447.

[24] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *12th USENIX Symp. Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA: USENIX Association, May 2015, pp. 395–408.

[25] W. Allcock, J. Bresnahan, R. Kettimuthu, and M. Link, "The Globus striped GridFTP framework and server," in *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, Nov. 2005, pp. 54–54.

[26] D. Barrera, L. Chuat, A. Perrig, R. M. Reischuk, and P. Szalachowski, "The SCION Internet architecture," *Commun. ACM*, vol. 60, no. 6, pp. 56–65, May 2017.

[27] B. Töpel, M. Karlsson, A. Duyck, *et al.*, *AF_XDP — the Linux kernel documentation*. [Online]. Available: https://www.kernel.org/doc/html/latest/networking/af_xdp.html.

[28] Anapaya Systems. "Anapaya network map." (Jan. 16, 2023), [Online]. Available: https://www.anapaya.net/about#widget_1635938094318_tabs_1.

[29] J. Kwon, J. A. Garcìa-Pardo, M. Legner, *et al.*, "Scionlab: A next-generation internet testbed," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, IEEE, 2020, pp. 1–12.

[30] G. Vardoyan, N. S. Rao, and D. Towsley, "Models of tcp in high-bdp environments and their experimental validation," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, IEEE, 2016, pp. 1–10.

[31] J. Iyengar and M. Thomson, *QUIC: A UDP-based multiplexed and secure transport*, RFC 9000, May 2021. DOI: 10.17487/RFC9000.

[32] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Transactions on Networking*, vol. 24, no. 1, pp. 596–609, Feb. 2016.

[33] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proceedings of the 13th international conference on emerging networking experiments and technologies*, 2017, pp. 160–166.

[34] M. Gartner, J. Wagner, M. Koppehel, and D. Hausheer, "Xdp-accelerated packet processing on scion endhosts," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2022, pp. 1–9.

[35] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu, *iperf3*, version 3.1.3, Jun. 8, 2016. [Online]. Available: http://software.es.net/iperf/.

[36] Jesper Dangaard Brouer. "Introduction to: Xdp and bpf building blocks." (Jan. 3, 2023), [Online]. Available: https://people.netfilter.org/hawk/presentations/ebplane2019/xdp-bpf-building-blocks.pdf.

[37] Europen Commission Energy Department. "Europe's investment in the ITER fusion project: Mastering the power of the sun and the stars." (Apr. 13, 2018), [Online]. Available: https://ec.europa.eu/info/news/looking-back-europes-contribution-iter-over-last-ten-years-2018-apr-12_en.

[38] KREONET Co. "KREONET." (Mar. 7, 2022), [Online]. Available: https://www.kreonet.net/.

[39] N. Jay, T. Gilad, N. Frankel, *et al.*, *A pcc-vivace kernel module for congestion control*, 2018.

[40] M. Dong, T. Meng, D. Zarchy, *et al.*, "PCC vivace: Online-learning congestion control," in *15th USENIX Symp. Networked Systems Design and Implementation (NSDI 18)*, Renton, WA: USENIX Association, Apr. 2018, pp. 343–356.

[41] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, Apr. 2003.

[42] Cheng Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *IEEE INFOCOM 2004*, vol. 4, Mar. 2004, pp. 2490–2501.

[43] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *ACM Queue*, vol. 14, September-October, pp. 20–53, 2016.

[44] M. Hock, M. Veit, F. Neumeister, R. Bless, and M. Zitterbart, "TCP at 100 Gbit/s – tuning, limitations, congestion control," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, Oct. 2019, pp. 1–9.

[45] B. Tierney and N. Hanford, "Recent Linux TCP updates, and how to tune your 100G host," in *Internet2 Technology Exchange*, Sep. 2016.

[46] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *SC '00: Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, Nov. 2000, pp. 38–38.

[47] CERN. "Grid file access library." (2022), [Online]. Available: https://github.com/cern-fts/gfal2.

[48] Netsec ETHZ, Ovgu Magdeburg. "Hercules source." (May 1, 2023), [Online]. Available: https://github.com/netsec-ethz/hercules.