Evaluating Encryption Overhead and Offload Techniques in High-Speed Data Transfers

Elie Kfoury*, Jose Gomez◇, Ali Mazloum*, Ali AlSabeh†, Jorge Crichigno*
*College of Engineering and Computing, University of South Carolina (USC)
◇Katz School of Business, Fort Lewis College (FLC)
†College of Sciences and Engineering, University of South Carolina Aiken (USCA)
Emails: {ekfoury, amazloum}@email.sc.edu, jagomez@fortlewis.edu,
ali.alsabeh@usca.edu, jcrichigno@cec.sc.edu

***Abstract*—The increasing demand for secure, high-speed data transfers in scientific and data-intensive applications presents a major challenge due to the significant processing overhead introduced by encryption. This paper evaluates the performance impact of encryption on GridFTP data transfers over a 100 Gbps link, focusing on throughput degradation, CPU utilization, and fairness in both encrypted and non-encrypted traffic scenarios. The experiments presented in this paper examine the influence of propagation delays and TCP Congestion Control Algorithms (CCAs), revealing that software-based encryption imposes severe CPU bottlenecks that limit scalability and link utilization. To address these limitations, this paper explores acceleration techniques that include CPU-based methods, such as Intel AES-NI, and hardware-assisted solutions, such as SmartNIC offloading. Results show that SmartNICs maximize throughput, achieving higher scalability compared to traditional approaches.**

***Index Terms*—GridFTP, SmartNIC, AES-NI, encryption overhead, high-speed data transfer, hardware offloading.**

I. Introduction

The transfer of large-scale datasets is increasingly common in scientific research and other data-intensive domains. In 2022 alone, the Department of Energy's ESnet, a dedicated network for scientific traffic, carried over 1.3 exabytes[1] of data to and from the sites it connects [1]. Such data volumes necessitate high-speed networks with links operating at 100 Gbps or higher to ensure efficient data movement.

GridFTP [2], a high-performance file transfer protocol built on top of the Transmission Control Protocol (TCP) [3], has become the *de facto* standard for transferring massive datasets across distributed environments. It offers several features to optimize data movement in high Bandwidth-Delay Product (BDP) Wide Area Networks (WANs), such as parallel stream transfers and tunable TCP buffer sizes [4–7]. These features have been shown to significantly enhance throughput over long distances. GridFTP is the protocol used by Globus [8], which is available in most research centers and universities.

With the increasing importance of data privacy and regulatory compliance in fields such as healthcare, genomics, and finance, encryption has become an essential component of modern data transfer protocols. GridFTP supports data channel encryption via the Grid Security Infrastructure (GSI), enabling secure transmission over untrusted networks. However, enabling encryption introduces substantial computational overhead. This overhead can lead to elevated CPU usage, reduced throughput, and under-utilization of available network capacity, forcing users to choose between performance and security [9, 10].

While previous studies have explored this trade-off, they typically focus on network speeds below 10 Gbps, which are no longer representative of current high-speed infrastructures. This paper addresses this gap by evaluating the performance implications of encrypted GridFTP transfers over a 100 Gbps link. The study examines throughput degradation, CPU utilization, and the impact of encryption under varying propagation delays and Congestion Control Algorithms (CCAs), including CUBIC and BBR. Additionally, the fairness between encrypted and non-encrypted flows is investigated under both equal and different start-time conditions, with and without network delay.

To better understand and mitigate the performance bottlenecks introduced by encryption, the paper evaluates software and hardware acceleration techniques. Specifically, it compares traditional software-based encryption, Intel's AES-NI instruction set, and SmartNIC-based cryptographic offloading [11]. The testbed consists of high-end servers directly connected through SmartNICs. The experiments emulate latency, which is important to perform realistic tests that match the current infrastructure. The results demonstrate that encryption imposes a significant bottleneck on high-speed transfers, particularly at scale. However, hardware-based solutions such as SmartNIC offloading can effectively overcome these limitations by reducing CPU load and maximizing throughput. These findings are essential for designing secure, scalable, and efficient data transfer solutions suitable for next-generation high-performance networks.

The remainder of this paper is organized as follows. Section II provides background on high-speed data transfers, GridFTP, and data channel encryption, along with a review of related work. Section III presents the experimental environment, including the testbed configuration, latency emulation methods, and measurement tools. Section IV presents a

Elie Kfoury and Jose Gomez contributed equally to this work.
[1]An exabyte is equal to 1,000 petabytes or 1 billion gigabytes.

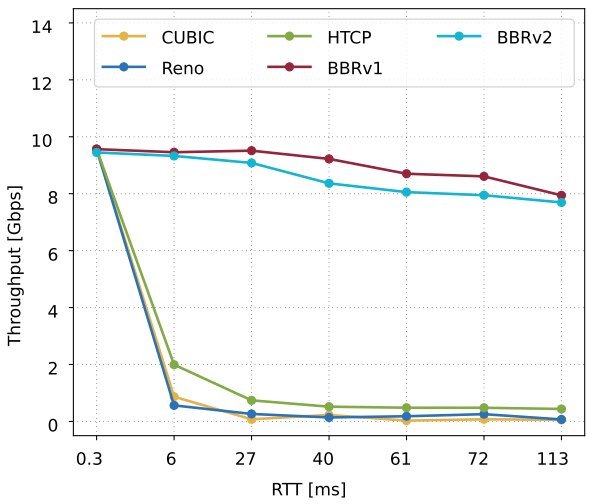


Fig. 1. Throughput performance of various TCP CCAs as a function of RTT over a 10 Gbps link. All algorithms experience throughput degradation as RTT increases, but to varying degrees. Traditional algorithms such as CUBIC, Reno, and HTCP exhibit steep declines in throughput, indicating high sensitivity to latency. In contrast, BBRv1 and BBRv2 maintain relatively stable performance, demonstrating better resilience in high BDP environments.

comprehensive evaluation of encryption overhead, analyzing CPU utilization, throughput degradation, fairness, and the influence of CCAs. Section V explores hardware-based acceleration using SmartNICs as a solution to mitigate encryption bottlenecks. Finally, Section VI summarizes key findings and outlines directions for future research.

II. Background and Related Work

This section provides a brief background on TCP, the issues of networks with large bandwidth-delay product, GridFTP, and data channel encryption. It then describes the related work.

A. The Role of BDP in High-Speed TCP Transfers

The Bandwidth-Delay Product (BDP) is an essential factor that characterizes the performance of TCP in Wide Area Networks (WANs) with high link capacities. It measures the data volume in transit, calculated by multiplying the network's bandwidth (BW) by its Round-Trip Time (RTT), defining the ideal TCP buffer size needed for maximizing throughput. Without properly adjusted buffers, networks with high bandwidth and latency may significantly underutilize available resources, causing inefficiencies. Fig. 1 illustrates how increased RTT significantly reduces throughput, even when packet loss rates remain low (i.e., 1 out of every 22,000 packets) over a 10 Gbps link. The results demonstrate that as RTT increases from 0.3ms to 113ms, the throughput drastically declines for all evaluated CCAs, including CUBIC, Reno, HTCP, BBRv1, and BBRv2. For instance, the throughput of algorithms such as CUBIC experiences notable reductions as RTT grows, highlighting their sensitivity to packet loss. In contrast, advanced algorithms such as BBRv1 and BBRv2 show improved resilience, maintaining

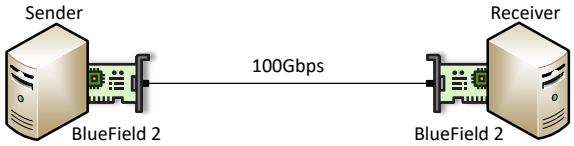


Fig. 2. Evaluations topology. BlueField-2 SmartNICs [14] are directly connected via a 100 Gbps link. Propagation delays are emulated on this link.

relatively higher throughput at elevated RTTs. This behavior underscores the critical impact of RTT on overall network performance.

B. GridFTP and Data Channel Encryption

GridFTP [2] is an extended version of the standard File Transfer Protocol (FTP), designed specifically for efficient, secure data transfers within Grid computing environments. GridFTP operates over TCP and enhances traditional FTP by introducing advanced features such as parallel streams, partial file transfers, and optimized network utilization through adaptive TCP buffer sizes. Furthermore, GridFTP integrates with the Grid Security Infrastructure (GSI), ensuring robust authentication and security mechanisms for data transmission.

A critical security aspect of GridFTP is Data Channel Encryption. Unlike standard FTP, which typically transmits non-encrypted data, GridFTP can optionally encrypt data channels using strong encryption methods provided by GSI. This ensures confidentiality and integrity of sensitive data during transmission, which is especially important in high-performance computing scenarios involving distributed, potentially untrusted networks. However, the literature [12, 13] has shown that enabling encryption features in GridFTP adds overheads that reduce the performance of data transfers.

C. Related Work

Previous studies have explored the trade-off between security and performance in GridFTP. Ohsaki *et al.* [15] evaluated the performance of GridFTP versions, highlighting that despite improvements in version 2, encryption still imposes noticeable overheads on data transfer performance. Vardoyan *et al.* [12] conducted an analysis of throughput bottlenecks introduced by enabling security features such as encryption and integrity protection in GridFTP. Their findings revealed significant performance degradation due to computational overhead associated with secure data transfers. Similarly, Rashti *et al.* [13] discussed the considerable overhead introduced by encryption and integrity checking in GridFTP, emphasizing that these security features are typically disabled to maintain transfer performance. They proposed hardware-assisted solutions to alleviate this overhead, demonstrating improved efficiency for secure, long-distance data transfers. Additionally, Poshtkohi and Ghaznavi-Ghoushchi [16] analyzed the overhead from context switching and parallel TCP streams, which further increases computational load when encryption is enabled. Lastly, Nadig *et al.* [17] conducted a comparative performance evaluation of high-performance data transfer tools, noting that GridFTP's security features,

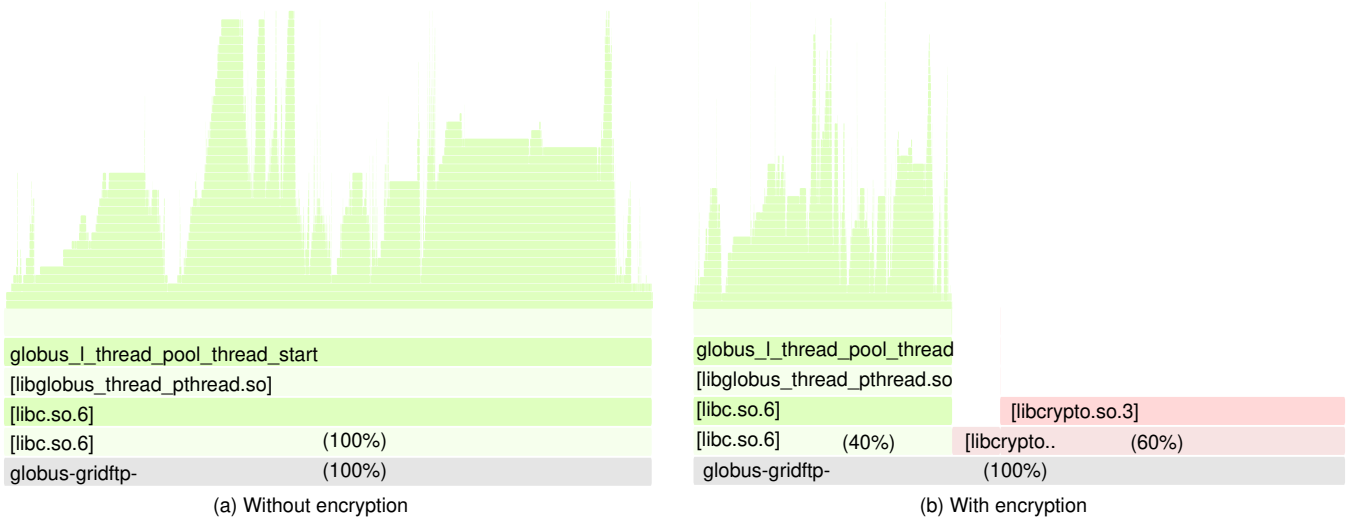


Fig. 3. Flame graph showing the share of processing used by encryption for a single data transfer using a single flow. (a) without encryption, 8.1 Gbps throughput; (b) with encryption, 0.51 Gbps throughput. 60% of processing is consumed on cryptographic functions during encrypted data transfer.

while beneficial, led to increased CPU utilization and reduced transfer speeds with encryption enabled.

While these studies provide valuable insights into the performance implications of secure GridFTP transfers, they are largely based on experiments conducted over network links no faster than 10 Gbps, making this study the first to evaluate GridFTP performance with security features enabled over a 100 Gbps link.

III. Experimental Setup and Results

The experimental topology is illustrated in Fig. 2. It consists of two servers directly interconnected via a 100 Gbps link. Each server is equipped with an NVIDIA BlueField-2 SmartNIC [14] and an Intel® Xeon® Silver 4214 CPU (32 cores at 2.20 GHz) with 16 GB of RAM. The Grid Community Toolkit is installed on both servers to reproduce data transfers.

Data Transfer Setup. To avoid I/O bottlenecks caused by disk read and write operations, data transfers used `/dev/zero` as the source and `/dev/null` as the destination files. Transfers were initiated using the `globus-url-copy` tool, a part of the Grid Community toolkit. Parallel streams were configured with the `-p` option. Data channel encryption was enforced using the `-dcpriv` option.

Latency emulation. Network delays are emulated using Linux Traffic Control (TC) with the Network Emulator (NETEM) module, combined with Intermediate Functional Block (IFB) devices on both servers to manage bidirectional traffic shaping. NETEM allows the injection of delays, packet loss, duplication, and reordering. Since TC only controls outgoing traffic, IFB devices are used to apply shaping to incoming traffic. In this setup, delay emulation is distributed across both servers. For instance, to emulate a total latency of 50ms, each server applies 25ms of delay on its respective IFB device. This way, the total delay experienced by packets traversing from one server to the other, and back, accurately reflects the intended 50ms RTT.

Metrics collection. As the experiments involve real data transfers using GridFTP, traditional tools like iPerf, which are designed for synthetic traffic generation and provide results only after completing a test, are not suitable. Instead, the results are passively collected using the Linux Socket Statistics (`ss`) tool. The `ss` tool provides detailed insights into active TCP connections, making it suitable for monitoring ongoing flows without interfering with the traffic. A custom script periodically parses the output of `ss -tin` to extract key performance metrics for each flow, such as RTT, congestion window (CWND), retransmissions, throughput rates, and more.

IV. Results and Evaluation

This section presents a performance evaluation introduced by encryption in high-speed data transfers using GridFTP over a 100 Gbps link. The evaluation focuses on four key aspects: CPU overhead, throughput degradation, fairness between encrypted and non-encrypted traffic, and the influence of propagation delays and CCAs. By comparing results across multiple configurations (i.e., varying the number of flows, parallel streams, and network latency), the study highlights how encryption imposes substantial computational burdens that limit scalability and efficiency. The following subsections detail the performance trends observed under each scenario and provide insights into the trade-offs involved when encrypting high-speed data transfers. Unless otherwise specified, the selected TCP CCA is CUBIC. This is because CUBIC is the default CCA used in Linux systems, and is recommended for data transfers in high-speed networks [18]. Also, the duration of the tests is 120 seconds, and the results are averages across 10 iterations. Additionally, the term “flow” denotes a data transfer; streams, on the other

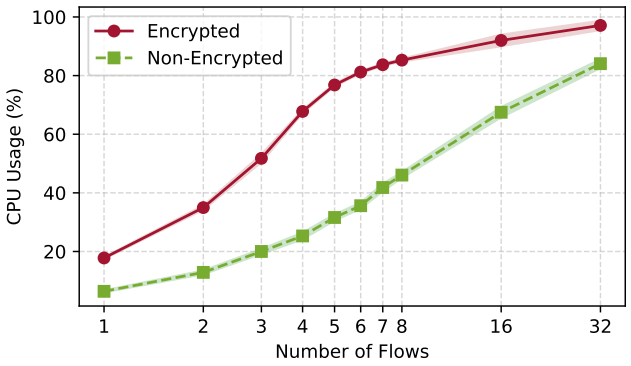


Fig. 4. CPU usage as a function of the number of flows for encrypted and non-encrypted data transfers. While CPU usage remains low in the non-encrypted case, encrypted transfers show significantly higher CPU utilization, approaching saturation at high flow counts due to cryptographic processing overhead.

hand, are the number of TCP connections (parallel streams) pertaining to each data transfer.

A. CPU Overhead Introduced by Encryption

Fig. 3 analyzes the CPU processing distribution during data transfers. It compares the overhead between a data transfer with and without encryption. The test initiates a single data transfer using one TCP stream. In the non-encrypted case depicted in Fig. 3(a), the transfer achieved a throughput of 8.1 Gbps, and the CPU cycles were spread across multiple system libraries and functions.

Conversely, enabling data channel encryption resulted in a significant drop in throughput to 0.51 Gbps. Consider Fig. 3(b). The flame graph reveals that approximately 60% of CPU processing was consumed by cryptographic functions, as indicated by the dominance of `libcrypto.so` in the processing stack. This illustrates the computational burden

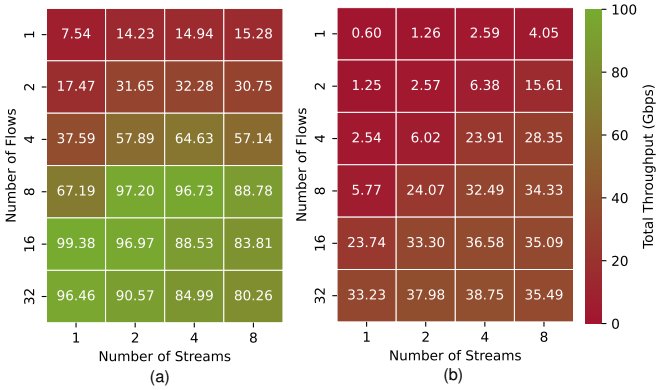


Fig. 5. Heatmap showing the total throughput (in Gbps) for various combinations of flow and stream counts. Each cell represents the average aggregate throughput achieved using the specified number of flows (y-axis) and parallel streams per flow (x-axis). (a) corresponds to transfers without encryption, while (b) shows encrypted transfers. As the number of flows and streams increases, throughput generally improves in both cases, although encrypted transfers saturate at lower values due to CPU overhead.

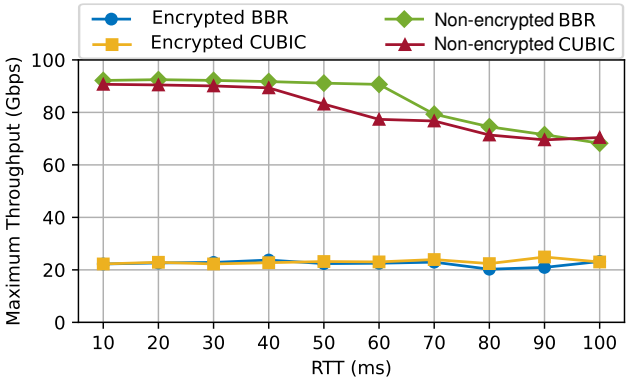


Fig. 6. Maximum throughput as a function of RTT for encrypted and non-encrypted transfers using BBR and CUBIC CCAs. While throughput declines with increasing RTT in all scenarios, the degradation is significantly more pronounced when encryption is enabled. BBR consistently outperforms CUBIC across the RTT range due to its design for high BDP environments. However, its performance advantage is constrained by the CPU overhead introduced by encryption, especially at higher latencies. These results demonstrate that encryption not only reduces overall throughput, but also amplifies the sensitivity of data transfers to RTT.

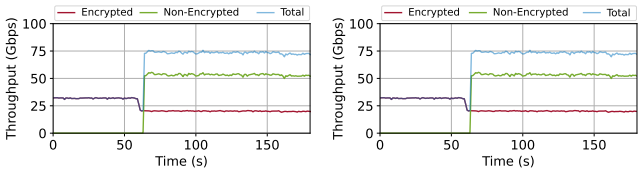
introduced by software-based encryption mechanisms, which directly impacts throughput and system efficiency.

Fig. 4 further illustrates the impact of encryption on CPU utilization by comparing encrypted and non-encrypted transfers across varying numbers of flows. In both cases, the CPU usage increases as the number of flows grows. However, encrypted transfers result in a significantly higher load—more than double that of the non-encrypted case—ultimately reaching nearly 100% utilization with 32 active flows.

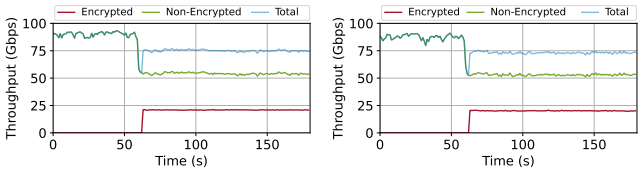
B. Encryption-Induced Throughput Degradation

Fig. 5 shows the total throughput for various combinations of flow and stream counts. Each cell represents the average aggregate throughput achieved using the specified number of flows (y-axis) and parallel streams per flow (x-axis). Fig. 5(a) corresponds to transfers without encryption, while Fig. 5(b) shows encrypted transfers. As the number of flows and streams increases, the throughput generally improves in both cases, although encrypted transfers saturate at lower values due to CPU overhead. The throughput of the encrypted traffic does not reach more than 40 Gbps over a 100 Gbps link. This result depicts that the link capacity is underutilized. Note that increasing the number of streams does not always guarantee higher throughput. For example, in the case of non-encrypted flows, throughput degrades when the number of streams becomes too large due to the overhead they introduce.

Fig. 6 shows the maximum throughput achieved as a function of the RTT using two CCAs: BBR and CUBIC. The test initiates eight data transfers with two parallel streams each. The results include both encrypted and non-encrypted transfers. As RTT increases, the throughput decreases. For encrypted transfers, performance remains similar regardless of the RTT or the CCA, as the bottleneck lies in the receiver's CPU. The throughput does not degrade with increasing RTT because multiple flows are active simultaneously. If packet

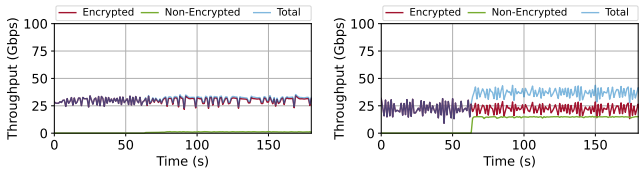


(a) CUBIC (b) BBR

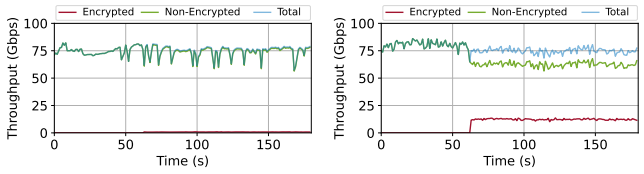


(c) CUBIC (d) BBR

Fig. 7. Throughput comparison between encrypted and non-encrypted flows under different start-order scenarios. (a) Encrypted traffic starts before non-encrypted traffic using the CUBIC congestion control algorithm. (b) Encrypted traffic starts first using BBR. (c) Non-encrypted traffic starts before encrypted traffic using CUBIC. (d) Non-encrypted traffic starts first using BBR.



(a) CUBIC (b) BBR



(c) CUBIC (d) BBR

Fig. 8. Throughput comparison between encrypted and non-encrypted flows under different start-order scenarios with a 50 ms round-trip delay. (a) Encrypted traffic starts before non-encrypted traffic using the CUBIC congestion control algorithm. (b) Encrypted traffic starts first using BBR. (c) Non-encrypted traffic starts before encrypted traffic using CUBIC. (d) Non-encrypted traffic starts first using BBR.

loss occurs in one flow, the others remain unaffected, allowing the aggregate throughput to stay high.

BBR outperforms CUBIC across the RTT range, consistent with its design for high BDP environments. Nevertheless, the benefits of BBR are limited when encryption is enabled, as the CPU overhead becomes the dominant bottleneck.

C. Fairness and Start-Time Effects Between Encrypted and Non-Encrypted Transfers

Fig. 7 explores whether the start order of encrypted and non-encrypted transfers affects the final performance outcome in a scenario without propagation delay. The results are presented for both CUBIC (Figs. 7(a) and 7(c)) and BBR (Figs. 7(b) and 7(d)). In the top row, Figs. 7(a) and 7(b), encrypted flows are initiated before non-encrypted flows. In the bottom row, Figs. 7(c) and 7(d), non-encrypted flows start first.

Regardless of which traffic type begins transmission first, both configurations converge to similar aggregate throughput values. This behavior is consistent across both CCAs. These results demonstrate that while encryption adds processing overhead and may influence short-term throughput dynamics, it does not result in long-term unfairness, when the RTT is small. The bandwidth distribution stabilizes quickly, confirming that the start order has minimal impact on overall performance.

D. Fairness and Start-Time Effects Between Encrypted and Non-Encrypted Transfers with Propagation Delay

Fig. 8 extends the previous fairness analysis by introducing a 50 ms round-trip propagation delay. The results are shown for both CUBIC (Figs. 8(a) and 8(c)) and BBR (Figs. 8(b) and 8(d)). In the top row, Figs. 8(a) and 8(b), encrypted flows are initiated before non-encrypted flows. In the bottom row, Figs. 8(c) and 8(d), non-encrypted flows start first.

Unlike the zero-delay scenario, total throughput varies significantly depending on whether encrypted or non-encrypted traffic starts first. When encrypted traffic starts first, the average total throughput for the CUBIC CCA is approximately 25 Gbps over the 180-second test duration. In contrast, when non-encrypted traffic starts first, the average total throughput reaches approximately 75 Gbps—a 3× increase. Additionally, fairness is heavily impacted; the traffic that starts first, whether encrypted or non-encrypted, continues to dominate the available bandwidth for the entire test duration.

The results differ for BBR. First, when encrypted traffic starts first, the total throughput of both encrypted and non-encrypted flows is higher than with CUBIC. Second, BBR improves fairness regardless of which traffic type starts first. Although the joining group of flows does not achieve perfect fairness with the existing flows, it does not starve either. Instead, it secures a reasonable share of the link—non-encrypted flows reach 20 Gbps after joining existing encrypted flows, while encrypted flows achieve 12 Gbps after joining existing non-encrypted flows.

It is worth noting that in real networks, where data transfers occur between geographically distant collaborators, propagation delays tend to be high. Additionally, since most networks use CUBIC, the performance of data transfers in the presence of encryption could be significantly impacted, requiring careful consideration.

V. Reducing Encryption Overhead with SmartNIC Offloading

Cryptographic operations are performed by the host CPU, creating significant bottlenecks in high-throughput scenarios, as demonstrated in previous experiments. To address these limitations, there is a growing need for offloading mechanisms that reduce computational overhead.

Intel's AES New Instructions (AES-NI) provide hardware-accelerated cryptographic processing, reducing CPU overhead

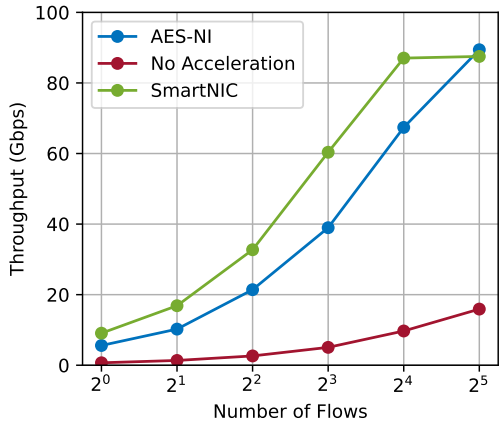


Fig. 9. Performance comparison of cryptographic processing: No-Acceleration (software only), AES-NI (CPU hardware acceleration), and SmartNIC Offload. SmartNICs offer the best throughput while freeing CPU resources for other tasks.

compared to purely software-based implementations [19]. However, despite these improvements, AES-NI still consumes substantial CPU resources.

SmartNICs offer a more effective solution by completely offloading cryptographic tasks from the host CPU to dedicated hardware. This approach achieves two key benefits: first, it significantly enhances throughput through parallelized cryptographic processing, and second, it frees up CPU resources for application-specific logic and other essential system tasks.

This experiment compares the throughput of three approaches: SmartNIC offload, AES-NI, and software-only encryption. A modified version of the iPerf3 tool—which supports kernel TLS (kTLS [20])—was used. When kTLS is enabled and the SmartNIC supports cryptographic offload, encryption can be offloaded, provided the chosen cipher (e.g., AES-256) is supported by the hardware.

As shown in Fig. 9, SmartNIC-based offloading outperforms both AES-NI and software-only encryption (No Acceleration). These results demonstrate that offloading cryptographic functions to SmartNICs enables higher throughput across all tested flow counts.

VI. Conclusion and Future Work

This paper presented an evaluation of the encryption overhead in high-speed data transfers over 100 Gbps networks using GridFTP. The experiments demonstrated that encryption significantly increases CPU utilization and limits throughput, especially as the number of parallel streams increases. These effects are further amplified with higher RTTs.

The paper concluded by exploring hardware-assisted acceleration techniques, including Intel's AES-NI and SmartNIC offloading. The results showed that SmartNICs performed the best, achieving higher throughput and reduced CPU load compared to the other approaches.

For future work, the authors plan to implement a module that enables kTLS and hardware offloading for GridFTP. This would improve the throughput when encrypted data transfers are present, reduces the CPU usage, and mitigates the fairness between encrypted and non-encrypted flows.

Acknowledgment

This work was supported by the U.S. National Science Foundation (NSF), awards 2417823 and 2346726.

References

[1] Energy Sciences Network (ESNet), "Accelerating World Changing Research Collaborations." [Online]. Available: https://tinyurl.com/95mnn36a, Accessed on 03-25-2025.

[2] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, and I. Foster, "Secure, efficient data transport and replica management for high-performance data-intensive computing," in *2001 Eighteenth IEEE Symposium on Mass Storage Systems and Technologies*, 2001.

[3] J. Postel, "RFC 793: Transmission control protocol (TCP)," September 1981.

[4] E. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Computer Communications*, 2020.

[5] J. Gomez, E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "A performance evaluation of TCP BBRv2 alpha," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020.

[6] J. Gomez, E. Kfoury, J. Crichigno, and G. Srivastava, "Understanding the performance of TCP BBRv2 using FABRIC," *2023 IEEE BlackSeaCom, Istanbul, Turkiye*, 2023.

[7] J. Gomez, E. Kfoury, J. Crichigno, and G. Srivastava, "Evaluating TCP BBRv3 performance in wired broadband networks," *Computer Communications*, vol. 222, 2024.

[8] Globus, "Globus, a Non-profit Service." [Online]. Available: https://www.globus.org, Accessed on 03-25-2025.

[9] A. AlSabeh, A. Mazloum, E. Kfoury, J. Crichigno, and H. Berry, "Enabling line-rate TLS SNI inspection in P4-programmable data planes," *IEEE/IFIP Network Operations and Management Symposium*, 2025.

[10] A. Mazloum, A. AlSabeh, E. Kfoury, and J. Crichigno, "Security applications in P4: Implementation and lessons learned," *Computer Networks*, vol. 257, 2025.

[11] E. Kfoury, S. Choueiri, A. Mazloum, A. AlSabeh, J. Gomez, and J. Crichigno, "A comprehensive survey on SmartNICs: Architectures, development models, applications, and research directions," *IEEE Access*, 2024.

[12] G. Vardoyan, R. Kcttimuthu, M. Link, and S. Tuecke, "Characterizing throughput bottlenecks for secure GridFTP transfers," in *International Conference on Computing, Networking and Communications (ICNC)*, 2013.

[13] M. Rashti, G. Sabin, and R. Kettimuthu, "Long-haul secure data transfer using hardware-assisted gridftp," *Future Generation Computer Systems*, vol. 56, 2016.

[14] NVIDIA, "BlueField-2 DPU." [Online]. Available: https://tinyurl.com/4zc7d5ez, Accessed on 03-25-2025.

[15] H. Ohsaki and M. Imase, "Performance evaluation of data transfer protocol GridFTP for grid computing," in *World Academy of Science, Engineering and Technology*, vol. 23, 2006.

[16] A. Poshtkohi and M. Ghaznavi-Ghoushchi, "DotDFS: A Grid-based high-throughput file transfer system," *Parallel Computing*, vol. 37, no. 2, 2011.

[17] D. Nadig, E. Jung, R. Kettimuthu, I. Fosterz, N. Rao, and B. Ramamurthy, "Comparative performance evaluation of high-performance data transfer tools," in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2018.

[18] ESnet FasterData, "Linux Tuning." [Online]. Available: https://fasterdata.es.net/host-tuning/linux/, Accessed on 03-25-2025.

[19] Intel, "Advanced Encryption Standard New Instructions (AES-NI)." [Online]. Available: https://tinyurl.com/4wwhsa8c, Accessed on 03-27-2025.

[20] Linux Kernel, "Kernel transport layer security (kTLS)." [Online]. Available: https://www.kernel.org/doc/html/latest/networking/tls.html, Accessed on 03-25-2025.