Enhance the multipath connection using AF_XDP sockets

Ba Que Le*, Mathias Santos de Brito*, Elena-Ramona Modroiu*, Marius Corici†, Thomas Magedanz*

* Technical University of Berlin, Berlin, Germany
{b.le, mathias.santos.de.brito, elena-ramona.modroiu, thomas.magedanz}@tu-berlin.de

† Fraunhofer FOKUS Institute, Berlin, Germany
{marius-iulian.corici}@fokus.fraunhofer.de

***Abstract*—Multipath is a technique that utilizes multiple links to establish an abstracted, unified connection for upper-level applications. This type of connection offers distinct advantages regarding overall bandwidth and reliability compared to regular single-path connections. Nevertheless, as the throughput and number of links increase, the associated overhead in path management and data processing increases to the extent that it may surpass the processing capacity of the CPU. This paper delves into enhancing multipath connections by leveraging Linux's XDP socket to process Tx and Rx traffic. By promptly separating the multipath traffic from the kernel's network stack with an XDP socket — a technology known for its high performance and efficiency, the connection will consume fewer resources and experience reduced latency. Furthermore, we present an implementation of this concept that can deliver a substantial increase in throughput, up to 146% when compared to connections created by link aggregation in round-robin mode, with further potential for improvement.**

***Index Terms*—Multipath, Link-Aggregation, XDP, AF_XDP, Backhaul, 5G, 6G**

I. Introduction

Enhancing network throughput for resource-intensive applications poses a formidable challenge. Currently, Ethernet and InfiniBand are the predominant communication technologies in data centers and high-performance computing (HPC) environments [1], offering line capacities ranging from 100-200Gbps [2] [3]. While these bandwidth figures are impressive in the network connection field, they fall short when compared to the internal processing capability of a single computer system. For instance, AMD's "Infinity Fabric" bridge delivers in-package bandwidth of 256GB/s on the Zeppelin SoC family [4] and up to 800GB/s in peer-to-peer tests between GPUs using multiple bands [5]. This indicates that the connection between computer instances can become a bottleneck for distributed, data-intensive use cases, such as data centers or telecommunication. In such usage scenarios, a multipath connection could offer distinct advantages by providing an abstracted connection that conceals the intricate details of combining links from the application. Consequently, it facilitates horizontal scaling of network capacity and can serve as an intermediate solution to overcome forthcoming physical and economic limitations.

Numerous multipath (MP) solutions are available for selection. One such option is Link Aggregation (LAG, often referred to as *bonding*), which operates as a low-level multipath implementation. It manages the link's flows at the MAC (Media Access Control) level and presents the user with a unified, virtual network interface. However, this solution only offers limited and unordered enhancement in single-flow throughput compared to the total link capacities [6].

Alternatively, some solutions rely on the kernel's network stack to exchange traffic, including GridFTP (TCP) and MPTCP (TCP subflows) [7] [8]. However, Linux networking, designed as a general-purpose software stack, faces limitations in handling multi-gigabit connections. The achievable throughput — slightly exceeding 40Gbps per CPU core — depends on factors such as the type of flows, software optimizations, and hardware [9]. Packet processing is influenced by several sources of overhead and delay within the network stack and the system, including data copying, buffering, interrupt handling, context switching, pre-emption [10] [11]. Consequently, multipath software that delegates the management of link connections to the Linux kernel inherits these limitations, along with additional overhead to process traffic across multiple links. This scenario poses a potential roadblock that could prevent a multipath solution from surpassing a certain throughput threshold.

There are several approaches to improving networking for Linux. The dual-kernel approach, involving real-time operating systems like RTAI or Xenomai, reduces significantly kernel latency by segregating real-time and standard tasks into different runtimes [12]. Less invasive, kernel patching approach enhances kernel's interrupt handling by making modification within the kernel code. However, the suitability of these approaches in production environments remains a subject of debate, considering factors such as compatibility, stability, and support [13].

A more commonly applied approach is kernel bypassing. Several mature projects employ this strategy, including

netmap, PF_RING, and DPDK. Kernel bypassing effectively diverts network flows from the kernel into a user-space network stack optimized for low latency and efficiency [14]. This often comes at the cost of removing the kernel's hardware NIC management, high resource usage, and the need to manage a sizable library and drivers [15].

AF_XDP represents an alternative "bypass" technique, although it operates distinctly. Leveraging on eBPF technology, the AF_XDP socket (often referred to as XDP socket or XDP) is a native Linux socket capable of swiftly moving a substantial volume of packets to and from a userspace application. To achieve this, low-level memory access is made available to the socket. Sending packets can be done by writing raw Ethernet packets to a ring buffer accessible by the network driver, and incoming packets can be copied directly to user application, essentially circumventing (or *bypassing*) the entire network stack. Unlike other *bypass* methods, by using a eBPF program, developers have complete control over all RX packets at the interception point within the kernel space, which enables early intervention before permitting specific packets to proceed along their initial paths. AF_XDP offers features that are highly advantageous for an ideal multipath library [16] [17]:

- Seamlessly integrates with the network stack while preserving the kernel's management of hardware NICs.
- Delivers scalable, high performance without compromising efficiency.
- Has no monopoly over the NICs.
- Low-level tasks run in isolated eBPF environment.

This paper presents the idea of enhancing multipath connection by utilizing XDP sockets to address overhead caused by large flow and multiple links. We have developed a prototype stack, which comprised of a multipath library and a pair of sender-comsumer application, to facilitate packet exchange between two separate machines via the library's multipath connection. The multipath packets are encapsulated in UDP datagrams. Preliminary findings reveal an utilization of over 88% of the total link's throughput, surpassing the over 72% achieved by Linux's Link Aggregation in Round-Robin mode.

One use case and motivation for the solution presented in this paper is multipath utilization in 5G networks' backhauls, especially in nomadic node scenarios. In some cases, the core is situated beside the Radio Access Networks, and multipath can enhance communication bandwidth between the RAN and the Core network. With the anticipated increase in bandwidth originating from User Equipment (UEs) in upcoming 6G networks, the backhaul should be capable of meeting the expected demand. On the other side 6G cores will require to be scalable to cope with this high throughput coming from the RAN, as demonstrated by innovative approaches like 6G Organic Core [18].

The rest of this paper is structured as follows: Section II presents the related work. Section III then outlines the problems with current multipath solutions and explains our motivation. Section IV describes the scope of this research, and Section V presents the prototype and test setup, followed by result and evaluation in Section VI. Section VII is dedicated to discussion and outlining future work. Finally, Section VIII concludes.

II. Related Work

Many multipath solutions use Linux's TCP and UDP stack to handle packet exchange. This includes MP-TCP, GridFTP (FTP extension), and MP-DCCP (extended-UDP). GridFTP has showcased its ability to deliver 27.3 Gbps over 30 Gbps links [19]. MP-TCP has demonstrated the capability to sustain a single-flow connection of 51.8 Gbps, achieved through the amalgamation of 6x10 Gbps links. However, sustaining this level of performance depends on specific kernel configurations and other optimization techniques, which may not be adequate for handling multiple links, each with more than 40 Gbps throughput [9] [20].

Modern network acceleration methods such as DPDK and XDP socket can be used to improve Linux networking. DPDK's performance is commonly regarded as the performance benchmark, often serving as the reference point for comparison [16]. However, the demands placed by this library are substantial, including detaching the NIC from the system and allocating dedicated resources. Additionally, the library itself is a large software project that requires learning and integration efforts, which we observed in the transition from DPDK to XDP for the Fraunhofer FOKUS NGNI's Open5GCore project [21]. This positions the AF_XDP socket as an appealing alternative since the socket is deeply ingrained within the kernel, where packets undergo isolated processing in kernel space [17], which means fewer dependencies and minimal re-implementation will be required.

We've also found that Hercules-SCION is a multipath solution that utilizes XDP technology. The authors presented Hercules as the primary transfer tool for the SCION network architecture [22] to enhance SCION's packet processing and dispatching performance. The tool leverages SCION's path control to forward traffic across multiple paths and has been demonstrated to surpass GridFTP in throughput tests [23] [24]. Although Hercules and our concept share many similarities, we have observed some notable differences. Firstly, Hercules is designed to work with SCION, a modern network architecture primarily focused on global-scale routing and management [25]. In contrast, our concept is designed purely to facilitate multipath connections between two machines. Secondly, the objectives of the concept encompass simplicity and portability, preferably in the form of a software library or module, rather than a complex software stack or a new architecture as the one used by Hercules.

III. Problem Statement

This work is motivated by the following:

Performance: Relation between Throughput and Packet per second. While throughput, which measures the amount of data transferred per second, often takes the spotlight in discussions, the ability to handle a high volume of packets per second is frequently overlooked. TCP protocol performs optimally with large data files to reach its full potential. Under unfavorable

conditions, i.e., when the flow consists of small files, performance can suffer considerably from a known issue called the ”lots of small files” (LOSF) [7]. This issue is explained that in most cases, the *per-byte costs* is dominated by the *per-packet costs*, which is caused by the high number and cost of system calls, memory allocation procedures, interrupt handling, and other factors necessary for processing each packet [26]. Efficient packet handling methods can significantly enhance Linux’s processing capability, increasing it from 4.8Mpps to 24Mpps with XDP socket and approaching 100Mpps with DPDK [17] [27].

Additional processing power can be released if such acceleration methods, like XDP and DPDK, are employed in multipath connection. This is particularly relevant when dealing with high-capacity links, which will require substantial processing power to handle the volume of data, manage the interrupt events generated by millions of packets, and deliver meaningful data to the destination application.

Maintainance and Stability. While many existing network acceleration solutions have seen limited adoption, XDP and eBPF are being employed by major organizations: firewall, filter, load-balancer by Facebook [28]; monitoring and analytics by Netflix [29]; security and observation in Cilium Project [30]; to cite some. DPDK has been widely adopted in various projects, including pfSense, Open vSwitch/OpenStack, OpenFlow, and 5G UPF implementations [31] [32] [33]. This widespread usage ensures the longevity of these frameworks, along with ongoing fixes and updates. Compared to DPDK, AF_XDP has the advantage of being integrated directly into the Linux kernel, eliminating the need for complex external dependencies.

IV. Scope of Research

This research is primarily concerned with exploring the concept and development of a prototype for evaluating the impact of applying XDP as a solution for multipath communication. In this phase, the initial focus is on the application of enhancements in throughput, i.e., the scenario where the packets of a single, large flow are distributed over different paths. In a second phase, other multipath aspects such as load balancing, multi-streams, latency, and heterogeneous links will be addressed. The prototype stack will establish a multipath connection between two machines, allowing traffic encapsulated in UDP packets to flow from the producer application to the consumer program in another machine. For the purpose of comparison, we will assess the throughput of the multipath connection created by our prototype and Linux Link-Aggregation. Additionally, to enable reliable measurement of goodput, we implement a buffering-waiting mechanism (referred as *reorderer*) to deliver partially resequenced traffic to the consumer application.

V. Experimental Methodology

The prototype stack consists of an XDP-based MP library, a sender and a receiver application (or *user applications*), all run in userspace (Figure 1). The library is responsible for initiating the XDP sockets, establishing the connection, exposing the endpoints to the user application, and handling the data and traffic.

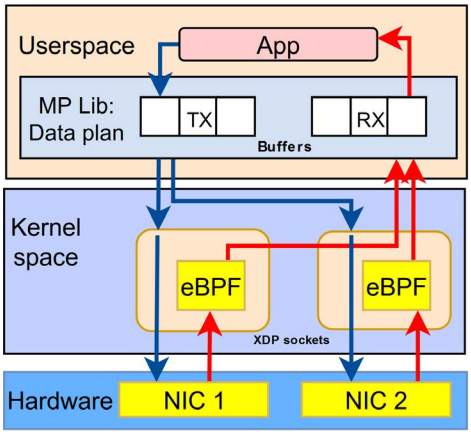

Fig. 1. Block diagram for the prototype stack with outgoing (TX) and incoming (RX) traffic

A transmission of a single packet commences at the sender machine, where the library accesses data authored by the sender application from a shared *mmap* memory region for subsequent processing. The processing procedure includes incorporating data into a multipath protocol header (as suggested by Krentz et al. [34]) and encapsulating the multipath packet within a UDP packet. After that, the UDP packets are ready for transmission over XDP sockets.

Upon reaching the receiver machine, a multipath raw Ethernet frame is screened by the XDP interceptor (referred to as *XDP kernel program*) and delivered to the library in userspace, where it undergoes a similar data processing procedure but in reverse order. Finally, the consumer application reads data from the shared *mmap* memory area, which was written to by the library.

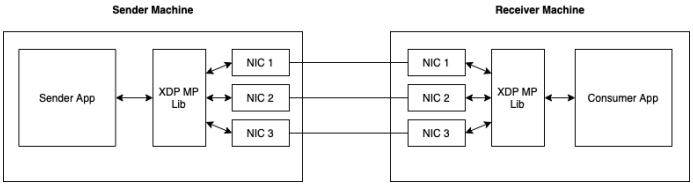

Fig. 2. Test setup

We use several header fields within the multipath header primarily for statistical purposes, although the multipath packet sequence number plays an important role in the reordering process. Since the prototype does not carry out any planning or scheduling, the outgoing packets are gathered, distributed, and received in batches across various links, resulting in an unpredictable reception order. An in-place Quick-sort strategy packet reorder implementation is therefore developed and integrated into the library to gather and rearrange received packages within a specified time frame. This procedure comprised of a buffer with 128 packet slots and a pre-defined

time interval, resembling, although less complex than, the one proposed by Amend et al. [35].

The sender and receiver machines are identical industrial-grade mini PCs with multiple NICs. These two machines are interlinked using Cat6 Ethernet cables (Figure 2). Each machine is equipped with Intel Atom Processor E3940 CPU and 4x Intel I210 1Gbps NIC, with one interface being dedicated to monitoring and control. For performance evaluation, we compare the goodput of our prototype with LAG's result operating in *balance-rr* (round-robin) mode since LAG works at roughly the same layer (Layer 2 Data Link layer). We configure two test scenarios, one using 2xNICs and the other using 3xNICs. In each scenario, we measure the goodput of the multipath connection established by either LAG or our prototype as follows:

- Single UDP flow over LAG connection.
- Single TCP flow over LAG connection.
- Multiple UDP flows over LAG connection.
- Multiple TCP flows over LAG connection.
- Single UDP-based flow over prototype's connection, reorderer deactivated.
- Single UDP-based flow over prototype's connection, reorderer activated.

Each scenario is executed five times. UDP payload length is set to 1024 bytes for the prototype. In *iperf3* tests conducted over LAG connections, we used the default parameters for UDP and TCP connections provided by *iperf3*. In case of multiple flows test, the number of flows is equal to the number of NICs under test in the current test scenario. All throughput values exclusively represent the payload size, excluding the headers.

VI. Experimental Result and Evaluation

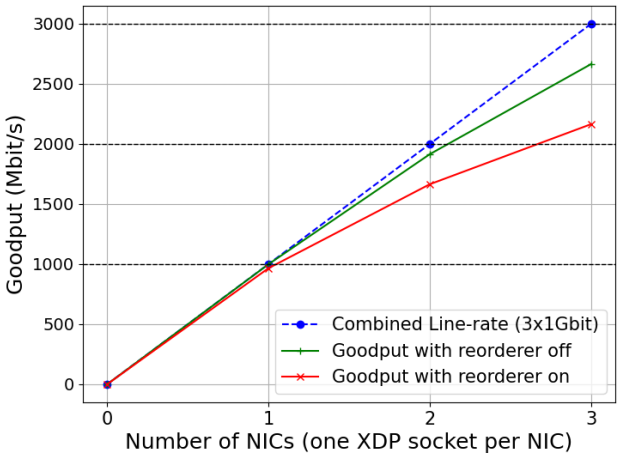

Fig. 3. Prototype's goodput with reorderer on and off compared to baseline

The library demonstrates the capability to deliver throughput that closely aligns with the total line rates to the consumer application. This is evident in Figure 3, where the green line represents the goodput measurements of the prototype stack. However, it is important to note that the consumer, by examining the packet sequence numbers, detects that more than 30% of the packets are not in the correct order. While enabling the packet reorderer effectively reduces such loss rate, it does introduce an additional processing load. We set a target for the packet out-of-order rate to be below 0.5% and then measure the average goodput of the connection, which yields slightly over 2Gbps, as depicted by the red line.

Moving forward, we compare prototype and LAG results. Figure 4 illustrates the goodput of the prototype through the red and green bars, while the values for single and multiple streams over LAG connection are represented respectively by the group of blue and brown bars. We note the following observations:

- Even with reordering activated, the prototype surpasses LAG's goodput using either UDP-TCP connections or single-multiple streams. In case of LAG's single UDP goodput, we record an increasement of 146% unordered goodput with our prototype (1081Mbps compared to 2665Mbps).
- Higher goodput can be attained using TCP rather than UDP connections. This could be explained by TCP's efficient bandwidth usage and congestion avoidance mechanism. UDP connections send packets with the best-effort style, while the TCP receiver informs the sender when and how much data should be transferred.

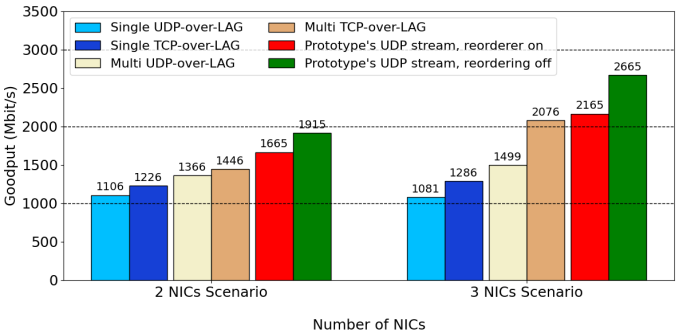

Fig. 4. Goodput comparison of link aggregation (balance-rr mode) and the XDP-based MP prototype

The LAG's *balance-rr* mode doesn't guarantee packet order. The tests reveal that when flows belonging to a UDP stream are distributed across multiple links, the *iperf3* server on the receiver side detects a significant number of out-of-order packets, ranging from 30% to 50% of the total packets. These values align closely with the library's out-of-order rate when the reorderer is disabled. Additionally, the LAG test results highlight the commendable performance of the Linux TCP stack in handling out-of-order packets and even surpassing UDP streams in terms of throughput. It's noteworthy that adding more streams doesn't consistently result in increased throughput. In most instances, optimal performance is achieved by using the same number of streams as the number of links.

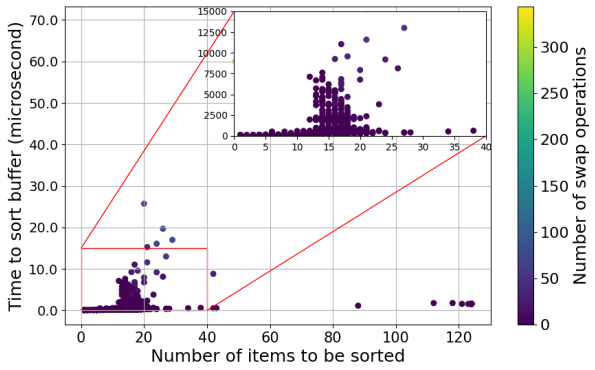

Fig. 5. Low traffic: Distribution of sort time, number of sorted items, and swap operations

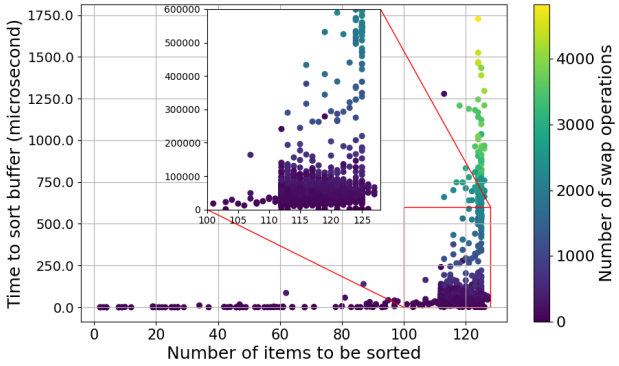

Fig. 6. High traffic: Distribution of sort time, number of sorted items, and swap operations

When the packet reorderer is active, we monitor the necessary swap operations and the time required to completely sort the buffer in low and high-traffic environment. In a session with a data transfer rate of 20MB/s (Figure 5), the reordering process frequently performs approximately 40 swap operations in less than 20 microseconds. During high-traffic sessions with a throughput of 150MB/s (∼1.2Gbps), the reorderer experiences significant stress (Figure 6). This leads to a substantial increase in both the number of required swap operations and the time it takes to perform them, resulting in elevated processing load and prolonged delays. This experiment indicates that, in order to maintain packet ordering, the prototype must consider one of two approaches: 1. Offloading the task to upper layer, similar to running TCP on top of LAG, which would contradict our goal of streamlining the packet processing path or 2. Employing common MP techniques such as scheduling, acknowledgments, queue management, flow partitioning, and burst (flowlets), as suggested in [36] [37] [38].

VII. Discussion and Future Work

Through the experiment, we notice that the reorderer is a significant performance bottleneck in our stack. Packet out-of-order in a multipath connection can be attributed to several factors: 1. Packets becoming scrambled across links at egress due to the absence of sender-receiver scheduling and cooperation, 2. Complications arising from heterogeneous links, and 3. Constraints imposed by buffer size and delay. Although a intelligent reorderer at the reception end can mitigate these issues to some extent, the complexity increases with the heterogeneity of individual links and higher throughput.

While still under evaluation, the prototype aims for simplicity and portability. The library can be packaged as a standalone library or a module with control and data planes to facilitate connection and management. The connection should be transparent to the user application and be able to exchange data with the applications through buffers or a virtual interface, such as TUN/TAP. Moreover, we are investigating the deployment of XDP's kernel program for critical multipath functionalities, as processing in kernel space introduces less overhead compared to our current implementation, which primarily resides in user space.

VIII. Conclusion

This paper introduced the concept of establishing a high-throughput multipath connection by employing XDP sockets to address system overhead. The evaluation revealed that our prototype, while built as a proof-of-concept, achieves a UDP throughput increase of up to 146% when compared to LAG on the same system. It's worth noting that this prototype has not been optimized and lacks specific XDP practices, leaving room for further enhancements. In the future, we aim to address practical challenges that have a significant impact on performance, such as reordering upon reception and sender-receiver scheduling.

While the results may not be directly comparable to more mature projects like GridFTP or MP-TCP, we argue that the concept holds substantial potential, especially considering the active development of XDP and its capability to scale effectively with additional CPU cores. XDP technology also offers attractive features for long-term projects, as it operates within the kernel, is inherently efficient, and avoids monopolistic restrictions on system resources. With further development, the concept could develop into a compelling multipath solution with ample room for future expansion.

Acknowledgment

The authors acknowledge the financial support by the German Federal Ministry for Education and Research (BMBF) within the project "Open6GHub" (TUB NUMBER, 16KISK003K).

References

[1] TOP500, "Highlights - June 2023," accessed November 12, 2023. [Online]. Available: https://www.top500.org/lists/top500/2023/06/highs/

[2] Ethernet Alliance, "Ethernet Roadmap," accessed November 12, 2023. [Online]. Available: https://ethernetalliance.org/wp-content/uploads/2023/03/EthernetRoadmap-2023-Website-REV-March-17.pdf

[3] InfiniBand Trade Association, "Infiniband Roadmap," accessed November 12, 2023. [Online]. Available: https://www.infinibandta.org/infiniband-roadmap/

[4] T. Burd, N. Beck, S. White, M. Paraschou, N. Kalyanasundharam, G. Donley, A. Smith, L. Hewitt, and S. Naffziger, ""Zeppelin": An SoC for Multichip Architectures," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, Jan. 2019, conference Name: IEEE Journal of Solid-State Circuits.

[5] AMD, "AMD Infinity Architecture," accessed November 12, 2023. [Online]. Available: https://www.amd.com/en/technologies/infinity-architecture

[6] IEEE 802.3 HSSG, "IEEE 802.3ad Link Aggregation: what it is, and what it is not," Published April 17, 2007. accessed November 12, 2023. [Online]. Available: https://www.ieee802.org/3/hssg/public/apr07/frazier_01_0407.pdf

[7] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, I. Foster *et al.*, "Gridftp pipelining," in *Proceedings of the 2007 TeraGrid Conference*, 2007.

[8] O. Bonaventure and M. Handley, "An Overview of Multipath TCP," 2012.

[9] Q. Cai, S. Chaudhary, M. Vuppalapati, J. Hwang, and R. Agarwal, "Understanding host network stack overheads," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 65–77. [Online]. Available: https://doi.org/10.1145/3452296.3472888

[10] N. Litayem and S. Ben Saoud, "Impact of the Linux Real-time Enhancements on the System Performances for Multi-core Intel Architectures," *International Journal of Computer Applications*, vol. 17, no. 3, pp. 17–23, Mar. 2011. [Online]. Available: http://www.ijcaonline.org/volume17/number3/pxc3872796.pdf

[11] C. Poellabauer, A. Schwan, and R. West, "Lightweight kernel/user communication for real-time and multimedia applications," in *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video - NOSSDAV '01*. Port Jefferson, New York, United States: ACM Press, 2001. [Online]. Available: http://portal.acm.org/citation.cfm?doid=378344.378365

[12] V. Yodaiken *et al.*, "The RTLinux Manifesto," in *Proc. of the 5th Linux Expo*, 1999.

[13] F. Reghenzani, G. Massari, and W. Fornaciari, "The Real-Time Linux Kernel: A Survey on PREEMPT_rt," *ACM Computing Surveys*, vol. 52, no. 1, p. 20, Feb. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3297714

[14] T. Barbette, C. Soldani, and L. Mathy, "Fast userspace packet processing," in *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. Oakland, CA, USA: IEEE, May 2015, pp. 5–16. [Online]. Available: http://ieeexplore.ieee.org/document/7110116/

[15] H. Redžović, A. Smiljanić, and M. Vesović, "Implementation and Performance Comparison of High-Capacity Software Routers," *Computer Networks*, vol. 183, p. 107585, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620312238

[16] M. Karlsson and B. Topel, "The path to dpdk speeds for af xdp," in *Linux Plumbers Conference*, 2018. [Online]. Available: http://vger.kernel.org/lpc_net2018_talks/lpc18_paper_af_xdp_perf-v2.pdf

[17] T. Høiland-Jørgensen, J. D. Brouer, D. Borkmann, J. Fastabend, T. Herbert, D. Ahern, and D. Miller, "The eXpress data path: fast programmable packet processing in the operating system kernel," in *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. Heraklion Greece: ACM, Dec. 2018, pp. 54–66. [Online]. Available: https://dl.acm.org/doi/10.1145/3281411.3281443

[18] Corici MI, Eichhorn F, Bless R, Gundall M, Lindenschmitt D, Bloessl B, Petrova M, Wimmer L, Kreuch R, Magedanz T, Schotten HD, "Organic 6G Networks: Vision, Requirements, and Research Approaches," *IEEE Access*, 2023.

[19] 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*, 2005, pp. 54–54.

[20] C. Paasch, G. Detal, S. Barré, F. Duchêne, O. Bonaventure, "The fastest TCP connection with Multipath TCP," 2015, accessed November 12, 2023. [Online]. Available: http://multipath-tcp.org/pmwiki.php/Main/50Gbps

[21] Scheich, Christian and Corici, Marius and Buhr, Hauke and Magedanz, Thomas, "eXpress Data Path Extensions for High-Capacity 5G User Plane Functions," in *Proceedings of the 1st Workshop on eBPF and Kernel Extensions*, 2023, pp. 86–88.

[22] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: a secure Internet architecture*. Springer, 2017.

[23] 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*. Budapest, Hungary: IEEE, Apr. 2022, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/9789923/

[24] M. Gartner, J.-P. Smith, M. Frei, F. Wirz, C. Neukom, D. Hausheer, and A. Perrig, "Hercules: High-Speed Bulk-Transfer over SCION," in *2023 IFIP Networking Conference (IFIP Networking)*. Barcelona, Spain: IEEE, Jun. 2023, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/10186366/

[25] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION: Scalability, Control, and Isolation on Next-Generation Networks," in *2011 IEEE Symposium on Security and Privacy*. Oakland, CA, USA: IEEE, May 2011, pp. 212–227. [Online]. Available: http://ieeexplore.ieee.org/document/5958031/

[26] L. Rizzo, "netmap: a novel framework for fast packet i/o," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 101–112.

[27] Intel, "Intel Ethernet's Performance Report with DPDK 22.11," published December 6th, 2022. accessed November 12, 2023. [Online]. Available: https://fast.dpdk.org/doc/perf/DPDK_22_11_Intel_NIC_performance_report.pdf

[28] Facebook, "Open-sourcing Katran, a scalable network load balancer," 2018, accessed November 12, 2023. [Online]. Available: https://engineering.fb.com/2018/05/22/open-source/open-sourcing-katran-a-scalable-network-load-balancer/

[29] Netflix, "How Netflix uses eBPF flow logs at scale for network insight," accessed November 12, 2023. [Online]. Available: https://netflixtechblog.com/how-netflix-uses-ebpf-flow-logs-at-scale-for-network-insight-e3ea997dca96

[30] M. A. Vieira, M. S. Castanho, R. D. Pacífico, E. R. Santos, E. P. C. Júnior, and L. F. Vieira, "Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–36, 2020.

[31] G. Pongrácz, L. Molnár, and Z. L. Kis, "Removing Roadblocks from SDN: OpenFlow Software Switch Performance on Intel DPDK," in *2013 Second European Workshop on Software Defined Networks*, Oct. 2013, pp. 62–67, iSSN: 2379-0369.

[32] Intel, "ZTE's High Performance 5G Core Network UPF Implementation," accessed November 12, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/wireless-network/zte-high-performance-5g-core-network-upf-paper.html

[33] NEC, "White paper: Nec's upf maximizes 5g value with high performance and flexibility in containerized, virtualized or physical deployments." [Online]. Available: https://www.nec.com/en/global/solutions/5g/download/pdf/NEC_UPF_Whitepaper.pdf

[34] K.-F. Krentz and M.-I. Corici, "Poster: multipath extensions for WireGuard," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–3.

[35] M. Amend, N. R. Moreno, M. Pieska, A. Kassler, A. Brunstrom, V. Rakocevic, and S. Johnson, "In-network Support for Packet Reordering for Multiaccess Transport Layer Tunneling," in *2022 IEEE 11th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*. Rome, Italy: IEEE, Nov. 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9963814/

[36] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 51–62, Mar. 2007. [Online]. Available: https://dl.acm.org/doi/10.1145/1232919.1232925

[37] J. Lin, X. Zhang, X. Gao, P. Kang, Y. Zhou, Y. Ouyang, and T. Feng, "Packet Reordering in the Era of 6G: Techniques, Challenges, and Applications," *Electronics*, vol. 12, no. 14, p. 3023, Jul. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/14/3023

[38] J. Huang, W. Lv, W. Li, J. Wang, and T. He, "QDAPS: Queueing Delay Aware Packet Spraying for Load Balancing in Data Center," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. Cambridge: IEEE, Sep. 2018, pp. 66–76. [Online]. Available: https://ieeexplore.ieee.org/document/8526805/