INSANE: A Uniform API for QoS-aware Host Networking as a Service

Lorenzo Rosa, Andrea Garbugli, Antonio Corradi, Paolo Bellavista
Department of Computer Science and Engineering
University of Bologna
Bologna, Italy
{name.lastname}@unibo.it

The ability to process and analyze data under stringent time constraints is quickly becoming a key requirement of modern data-driven applications, leading to a growing demand for systems that can achieve µs-scale latencies while maintaining high levels of reliability, scalability, and efficiency [3, 4]. To avoid becoming a bottleneck, widely popular systems, ranging from key-value stores to state-machine replication engines, have been carefully re-designed to leverage *kernel-bypassing* I/O techniques and modern hardware that implements common operating system services, such as host networking, in more efficient way. An even more radical approach to reduce service latencies is emerging under the paradigm of *edge cloud computing*, which combines such acceleration techniques with the idea of moving components on small-scale datacenter-like environments physically close to datasources, in support of latency-critical services (e.g., Multi-access Edge Computing platforms, or MEC, for 5G operators).

Although *kernel-bypassing* techniques coupled with modern hardware have proved effective to achieve time-sensitive data processing, two critical concerns still prevent their wide-scale adoption. On the one hand, existing techniques for datapath acceleration require the use of custom and usually low-level interfaces that make application development difficult and time-consuming, requiring advanced system expertise: not only applications must be re-architected and carefully optimized to fully leverage the performance benefits of acceleration technologies [1], but developers must also deal with the continuous release of updated device features and the concurrent deployment of different generations of hardware [2]. On the other hand, there are several possible techniques for host networking accelerations, which offload typical kernel tasks to userspace or even hardware: the Linux eXpress Data Path (XDP), the Data Plane Development Kit (DPDK), or Remote Direct Memory Access (RDMA). Each of these options defines its own programming abstractions, network access interface, and memory management: hence, application portability is very hard to provide, and *accelerated* services can be deployed only onto a single and highly homogeneous environment, whereas real-world platforms are usually quite the opposite, in particular in the so-called *cloud continuum* [3]. For instance, RDMA is only intermittently supported by major public clouds. Even more importantly, the increasingly popular cloud edge computing platforms are highly heterogeneous in terms of software and hardware resource availability, making such lack of portability even more troubling right where latency-aware applications are most needed.

We present INSANE (Integrated aNd Selective Acceleration for the Network Edge), a general-purpose and lightweight userspace network stack that provides a uniform API to a wide range of network acceleration technologies, including XDP, DPDK, and RDMA, even available to the same supported application, with the goal of easing the development, deployment, and portability of latency-critical services in the cloud continuum. Following a microkernel-inspired architecture [2], INSANE consists of two main components:

(1) A client library, exposing a uniform API with a minimal set of communication primitives, yet expressive enough to let developers define high-level and domain-specific abstractions on top of them. Through a set of *Quality of Service* (QoS) parameters, applications can define differentiated network requirements for their data flows, such as *latency-sensitiveness*, *reliability*, and *resource consumption*. Unlike in Demikernel [4], in our solution, flows with different requirements can be mapped to different technologies.
(2) A runtime, working as a *network stack as a service* for applications and offering common services for high-performance networking, including memory management for zero-copy transfers, efficient packet processing, and different packet scheduling strategies. A plugin-based architecture allows the specialization of such abstractions for each integrated network acceleration technology. In particular, the high-level QoS requirements specified by applications are used to dynamically map the flows to the most appropriate acceleration technology that is dynamically available at the deployment site.

Overall, INSANE significantly simplifies the development of latency-critical services based on *acceleration* techniques by offering *QoS-aware host networking as a service*, introducing minimal (ns-scale) overhead, and making applications portable across heterogeneous environments such as public clouds and edge cloud nodes, thus unleashing a new generation of ultra-low latency cloud continuum applications in several time-critical domains.

REFERENCES

[1] Anuj Kalia, Michael Kaminsky, and David G Andersen. 2016. Design guidelines for high performance RDMA systems. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. 437–450.
[2] Michael Marty, Marc de Kruijf, Jacob Adriaens, Christopher Alfeld, Sean Bauer, Carlo Contavalli, Michael Dalton, Nandita Dukkipati, William C Evans, Steve Gribble, et al. 2019. Snap: A microkernel approach to host networking. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 399–413.
[3] Jose Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. 2021. Towards low-latency service delivery in a continuum of virtual resources: State-of-the-art and research directions. *IEEE Commun. Surveys & Tutorials* 23, 4 (2021), 2557–2589.
[4] Irene Zhang, Amanda Raybuck, Pratyush Patel, Kirk Olynyk, Jacob Nelson, Omar S Navarro Leija, Ashlie Martinez, Jing Liu, Anna Kornfeld Simpson, Sujay Jayakar, et al. 2021. The demikernel datapath os architecture for microsecond-scale datacenter systems. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 195–211.