ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

This Is INSANE!

Kernel-Bypass Networking Finally Made Easy

Lorenzo Rosa

Department of Computer Science and Engineering
University of Bologna, Italy

December 15th, 2025
26th ACM/IFIP International Middleware Conference - Nashville, TN, USA

Introduction

- Postdoctoral researcher at the University of Bologna

- Specializing in cloud/edge computing, serverless systems, middleware, and industrial networking, with a focus on QoS and host-network performance.

- Main designer and developer of the **INSANE middleware**, subject of today's tutorial.

Email: lorenzo.rosa@unibo.it
Website: https://www.unibo.it/sitoweb/lorenzo.rosa/

Lorenzo Rosa
University of Bologna, Italy

Agenda

1. **Context, Motivation, Background (~ 45 min)**

Motivation & Overview of modern acceleration stacks (SmartNICs, RDMA, DPDK, XDP), their evolution, and the challenges of using them for portable applications.

2. The INSANE Middleware (~ 45 min)

Architecture, goals, and design of INSANE. Includes relevant related work and design rationale.

Coffee Break

3. Hands-On Session with INSANE (~ 90 min)

Guided exercise using a NATS-based application whose communication layer is incrementally replaced with INSANE.

Info and material

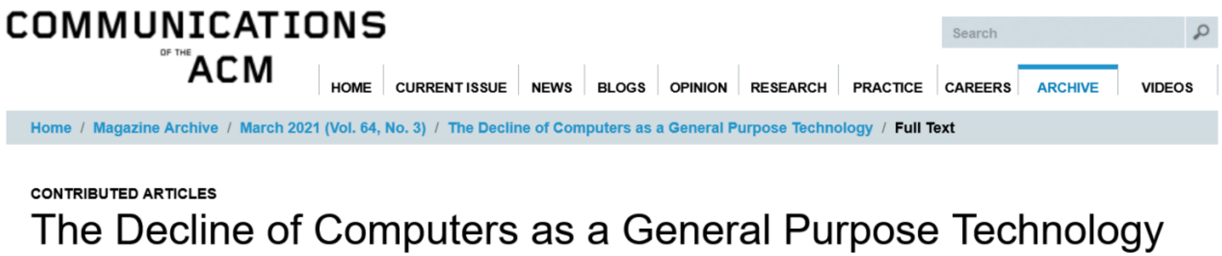
- Web page of the tutorial: http://insane-tutorial.ing.unibo.it
  - All the slides
  - Pointers to the repositories
  - Additional material

- Github repository for this tutorial: https://github.com/ellerre/insane-tutorial

- *INSANE middleware* repository: https://github.com/MMw-Unibo/INSANE

Introduction

General-purpose computing


COMMUNICATIONS OF THE ACM

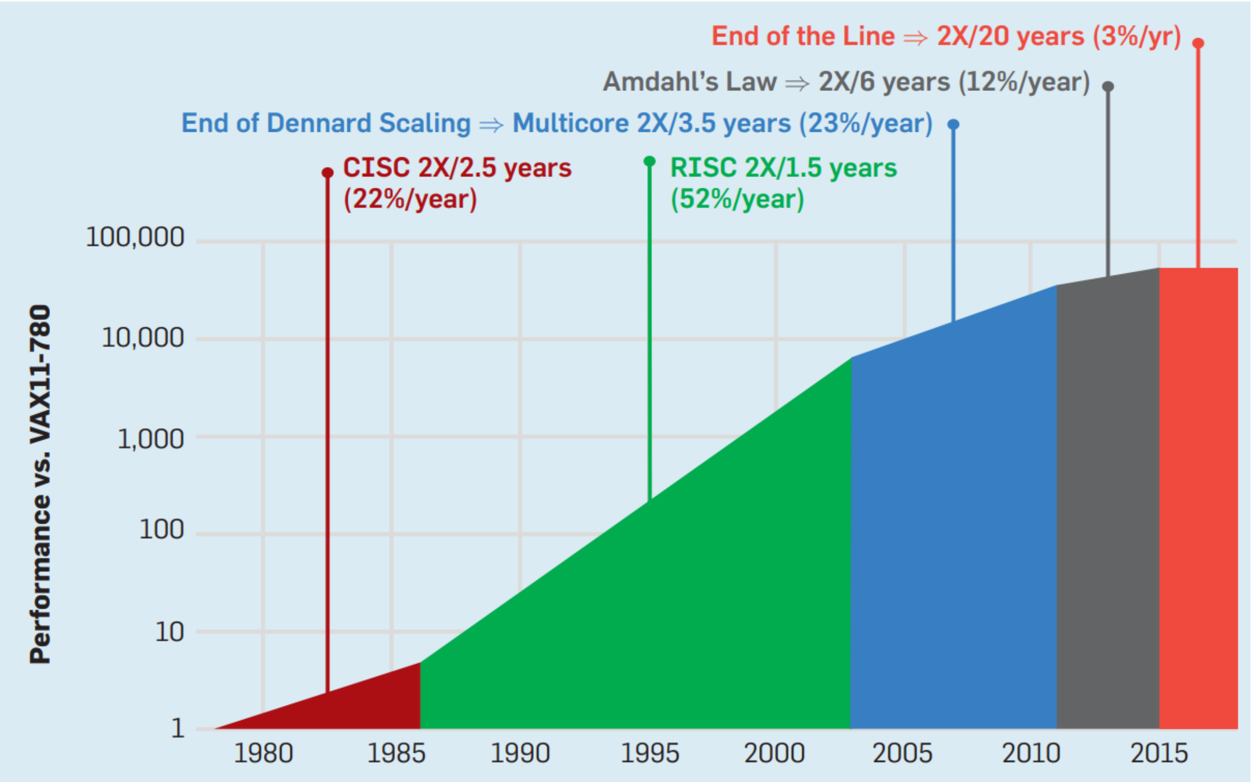
Search

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE | CAREERS | ARCHIVE | VIDEOS

Home / Magazine Archive / March 2021 (Vol. 64, No. 3) / The Decline of Computers as a General Purpose Technology / Full Text

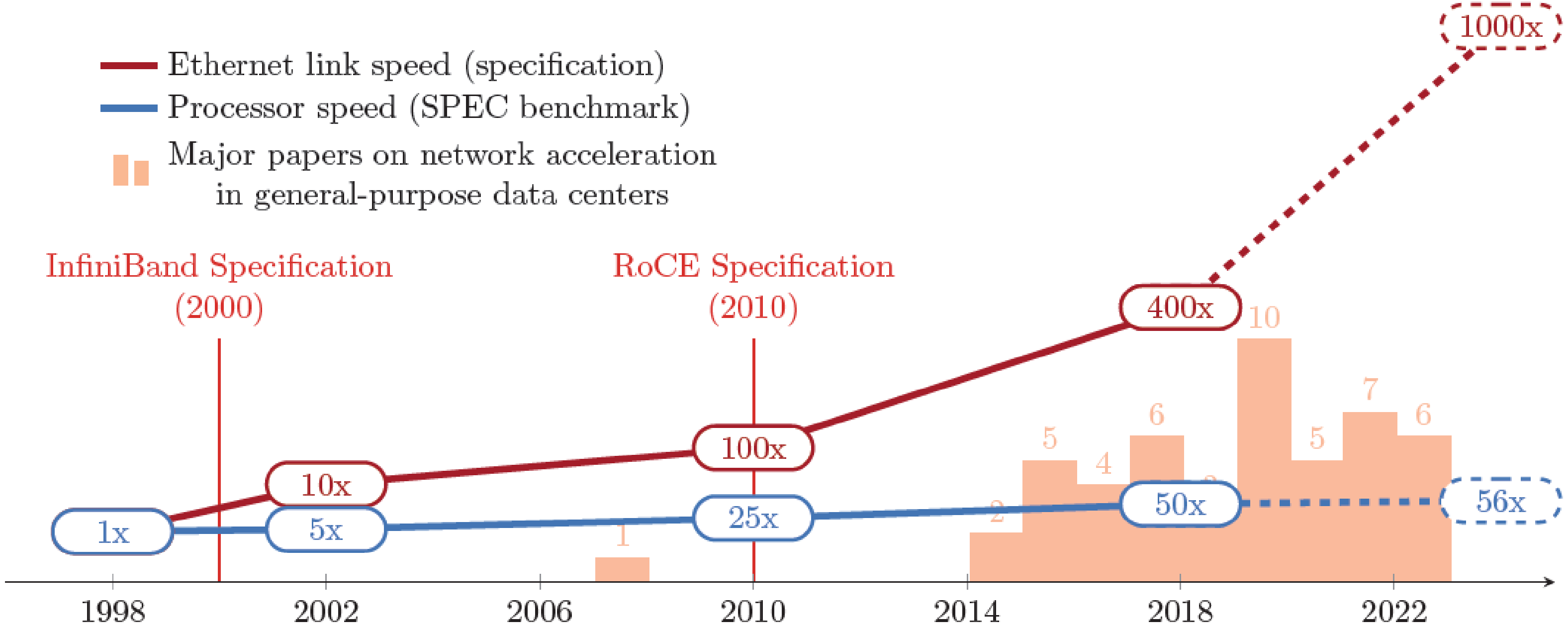
CONTRIBUTED ARTICLES

The Decline of Computers as a General Purpose Technology



Slow improvements lead to specialization

A new golden age: toward specialization

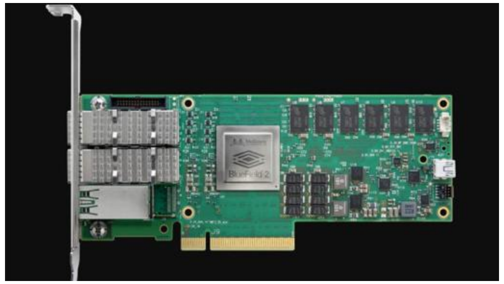


DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

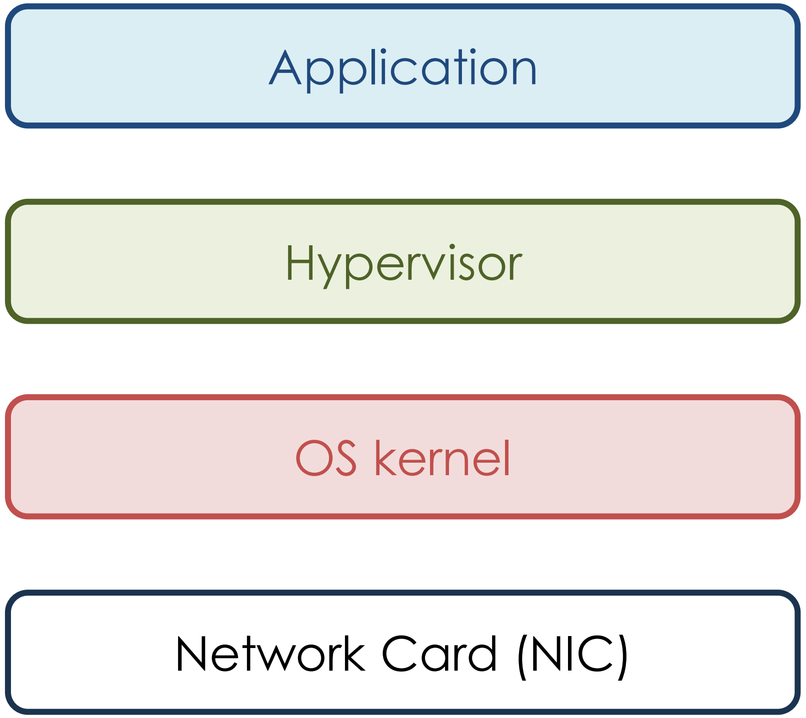


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

What about software?

Software components are designed for **general-purpose** architectures:

- Operating Systems
- Hypervisors
- Applications
  - user applications
  - middleware services



Software systems must evolve to accommodate the emerging specialized architectures

General-purpose networking becomes a bottleneck

Operating Systems and Hypervisors were designed under the assumption:

<u>I/O was slower than CPU processing</u>

Specialized hardware **reverses** this assumption:

<u>I/O is *much faster* than CPU processing</u>

Q1. How can we leverage the performance of modern networks and accelerators?

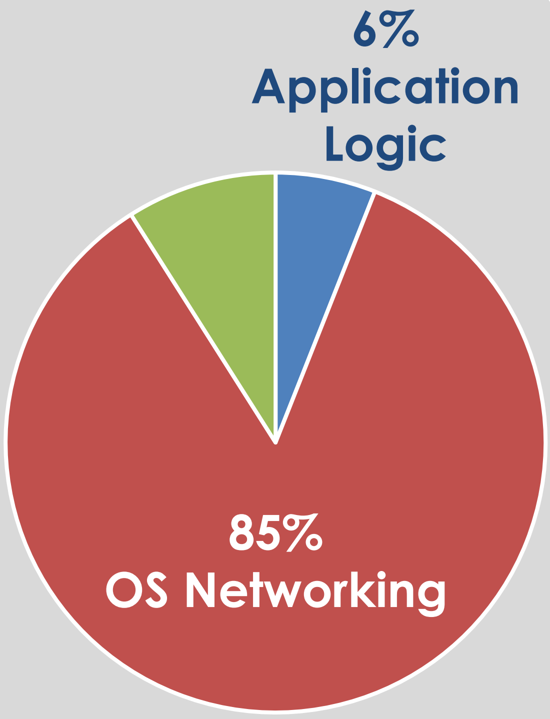
Q2. Can we preserve application portability?

CPU cycles breakdown

for an

Average K/V store request

TAS [EuroSys'19]



Kernel-Bypass Networking

Kernel-bypass approach: completely remove the OS from datapath

Application
↕
Hypervisor
OS kernel
↕
Network Card (NIC)

- data copies
- context switches
- inefficient protocol processing

Application
↕
Network Card (NIC) or Hardware Accelerator

- ✓ zero-copy transfers
- ✓ no context switches
- ✓ simplified protocol processing

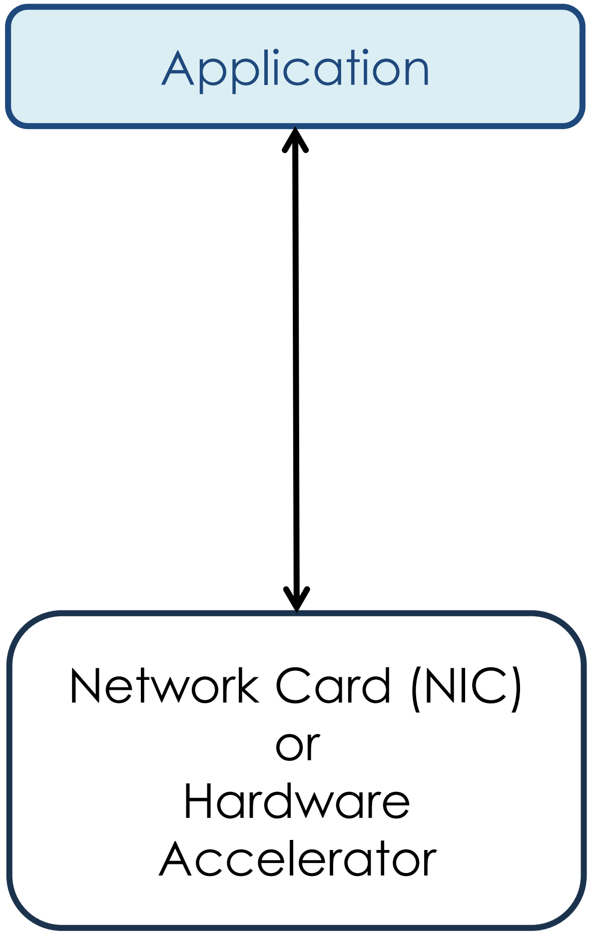
Kernel bypass improves I/O at the expenses of generality, portability, and virtualization

Kernel-bypass approach

The kernel-bypass approach enables developers to **fully leverage** the performance of modern specialized hardware.

There are many flavors of kernel-bypassing, but they all follow three key principles:

- **zero-copy** data transfers: reduce memory copy
- minimal **context switches**: improve cache efficiency & CPU usage
- **asynchronous** data processing: involve the CPU on the control plane, leave the data plane to the hardware.

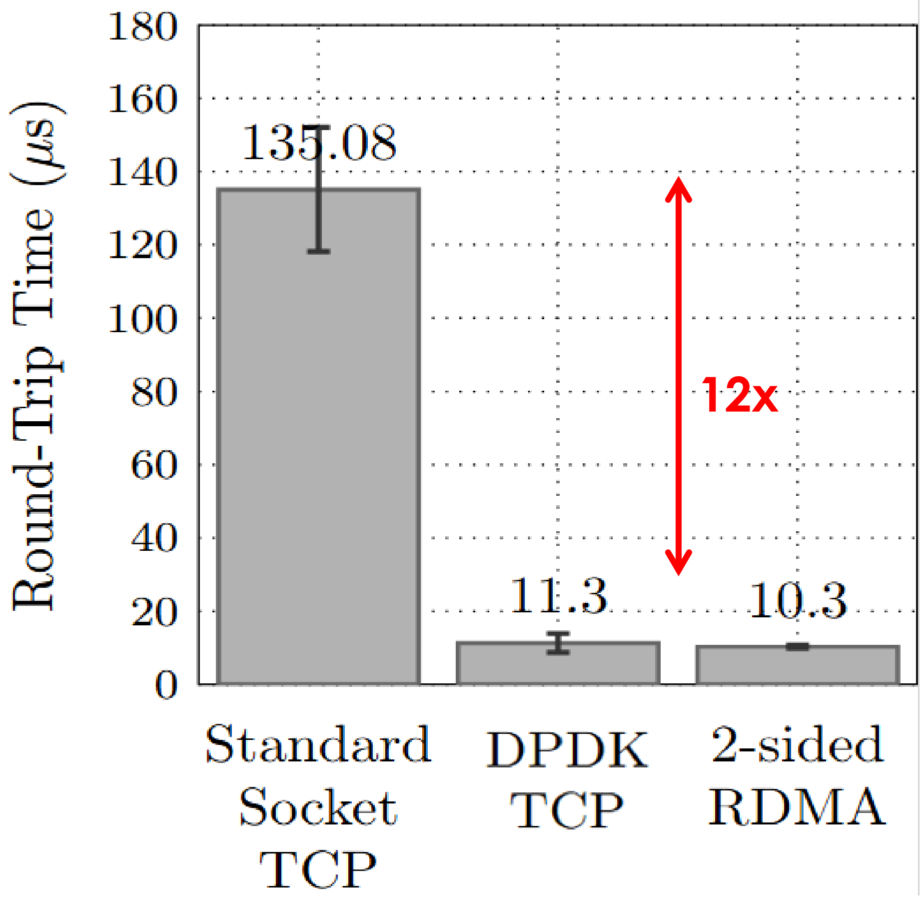


Kernel-bypass approach potential: example

Example: TCP echo server, 64 bytes messages

Testbed: an 8-core VM attached to a 200 Gbps NVIDIA ConnectX-6 (passthrough)

Kernel-bypassing can provide orders-of-magnitude improvements over standard OS-based networking.



ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

Several options available today for high-performance I/O

Hardware offloading has the highest potential, but:

- not always available
- raises security concerns with multi-tenancy
- can be wildly heterogeneous

RDMA

Programmable hardware

Software-based alternatives offer lower performance, but

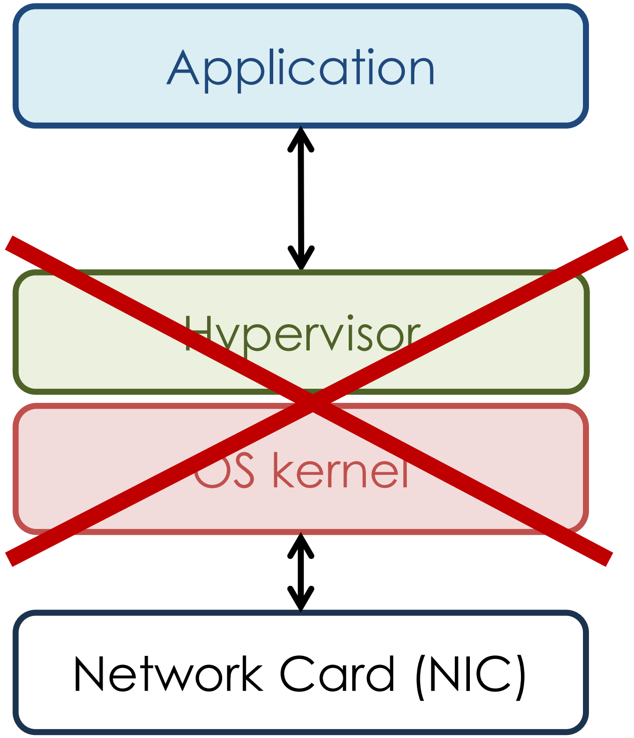
- still faster than OS-based networking
- usually available on commodity hardware

DPDK

Linux XDP

Rosa et al., Empowering Cloud Computing with Network Acceleration: A Survey. *Communications Surveys & Tutorials*, 2024

Kernel-bypass drawbacks: complexity and heterogeneity



When we remove the kernel:

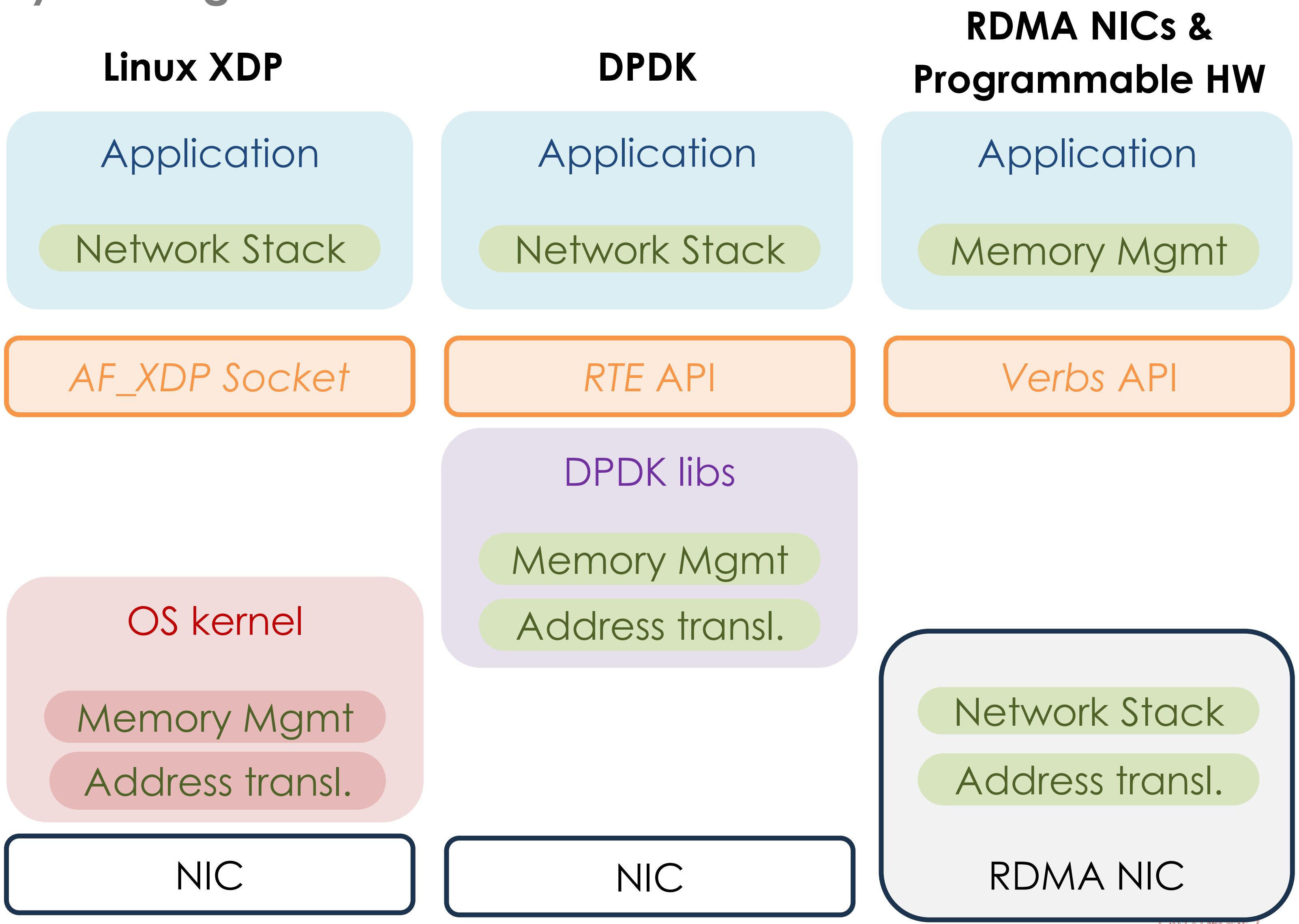
1. Which interface is exposed to the application?
2. Who implements the network stack (TCP/IP)?
3. Who manages memory management, address translation, etc?

Different kernel-bypass techniques have different answers, **forcing applications to specialize** for one of them.

High-performance I/O is deeply heterogeneous

The complexity of this heterogeneous landscape is exposed to user applications

1. Complex & Low-Level APIs
2. Scattered OS services



Kernel-bypass drawbacks: complexity and heterogeneity

Kernel-bypassing approach require:

- Application re-writing, harming **portability**.
- Complex and low-level API, requiring **coding expertise**.
- Userspace re-implementation of typical OS services, leading to **code duplication**.

The **average cloud users** cannot afford this complexity, despite the potential performance advantages.

High-performance I/O is not yet available *as a service* to cloud applications

Example: a "hello world" application

A simple **C application** sending a "hello world" string from a client to a remote server requires:

31 lines of code with kernel-based TCP

212 lines of code with DPDK + custom userspace TCP/IP stack (**~ 150,000 loc**)

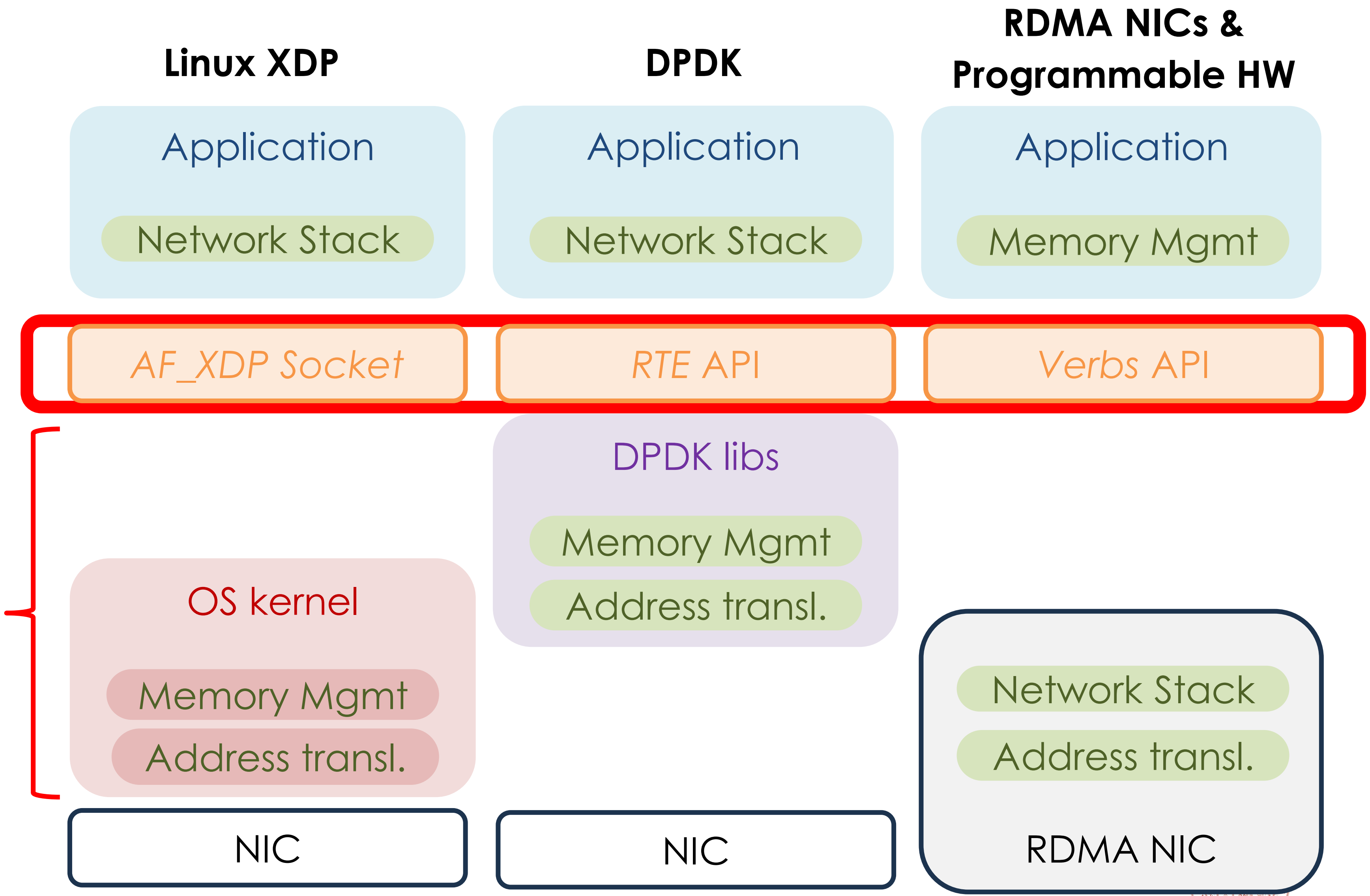
~200 lines of code with RDMA

The C code is low-level, requires **re-implementation** of services (e.g., memory management) and requires **careful optimizations** to yield maximum performance.

Which architecture for Network Acceleration as a Service (NAaaS)?

NAaaS: a general-purpose, high-perf datapath with:

1. Portable, high-level API
2. efficient OS features offered *as a service* to applications



A Middleware approach: existing solutions

Cloud apps already rely on **network middleware** to hide the complexity of communication among application components

NATS, ZeroMQ, MQTT, DDS, RabbitMQ, Kafka

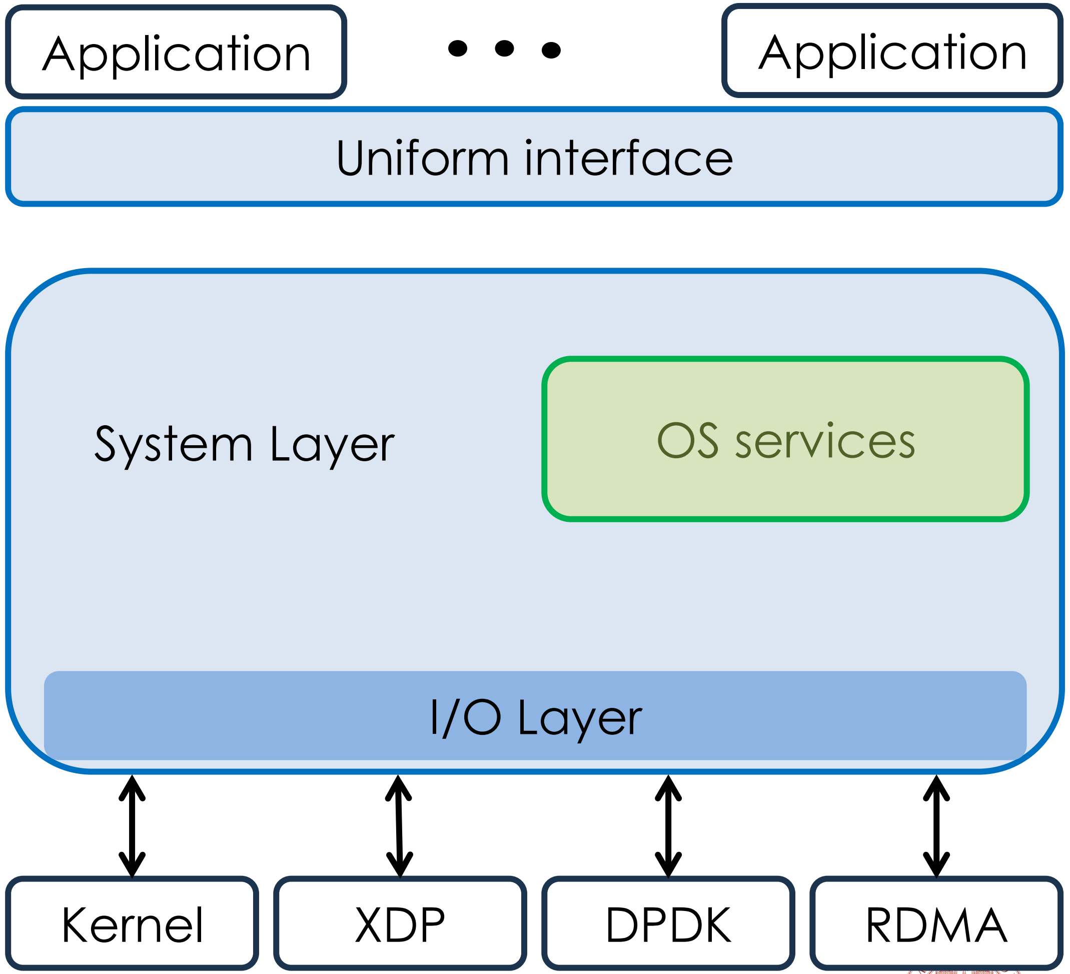
Those systems offer QoS options to control **semantic properties** of the communication (retransmissions, handshakes, etc.)

Their focus is on ease of deployment and portability; hence they **do not support** kernel-bypassing. They are not designed for the emerging high-performance networking hardware.

A Middleware approach to kernel-bypassing in the Cloud

A middleware approach for portable accelerated applications in the cloud should define:

1. An **interface layer** exposing a stack-agnostic set of primitives to user applications.
2. A **system layer** providing OS system services in a *centralized* fashion (OS-style), shared by multiple applications.
3. An **I/O layer** implementing network ops for each acceleration technology.



A possible approach: Library OS

A possible approach: put networking logic into application binaries (**library OS**).

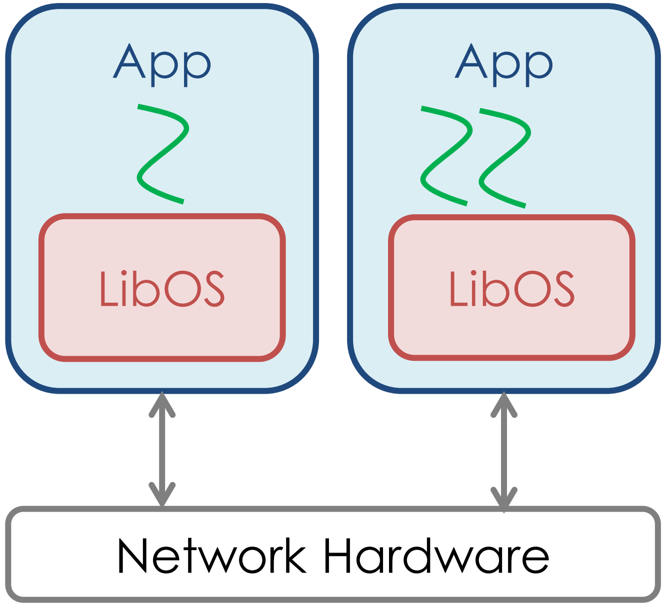
Explored by Demikernel [SOSP'21] to solve a similar problem and inspired by IX [OSDI '14], Arrakis [OSDI '14], and many others.

Advantages:

- No IPC required

Disadvantages:

- requires spin-polling in every application
- high communication setup time
- no centralization (i.e., scheduling)
- difficult to use in virtual environments (VMs, containers)



A sidecar approach to kernel-bypassing in the Cloud

A better approach for the cloud is the ***sidecar***, explored by TAS [EuroSys'19] or SNAP [SOSP'19] to solve a similar problem. More recently, Pegasus [EuroSys'25].

Disadvantages:

- IPC between apps and the datapath process

Advantages:

- centralized resource management
- data and instruction locality, resource efficiency
- scheduling and multiplexing options
- short communication setup time
- decoupled release cycles wrt applications