The INSANE middleware

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

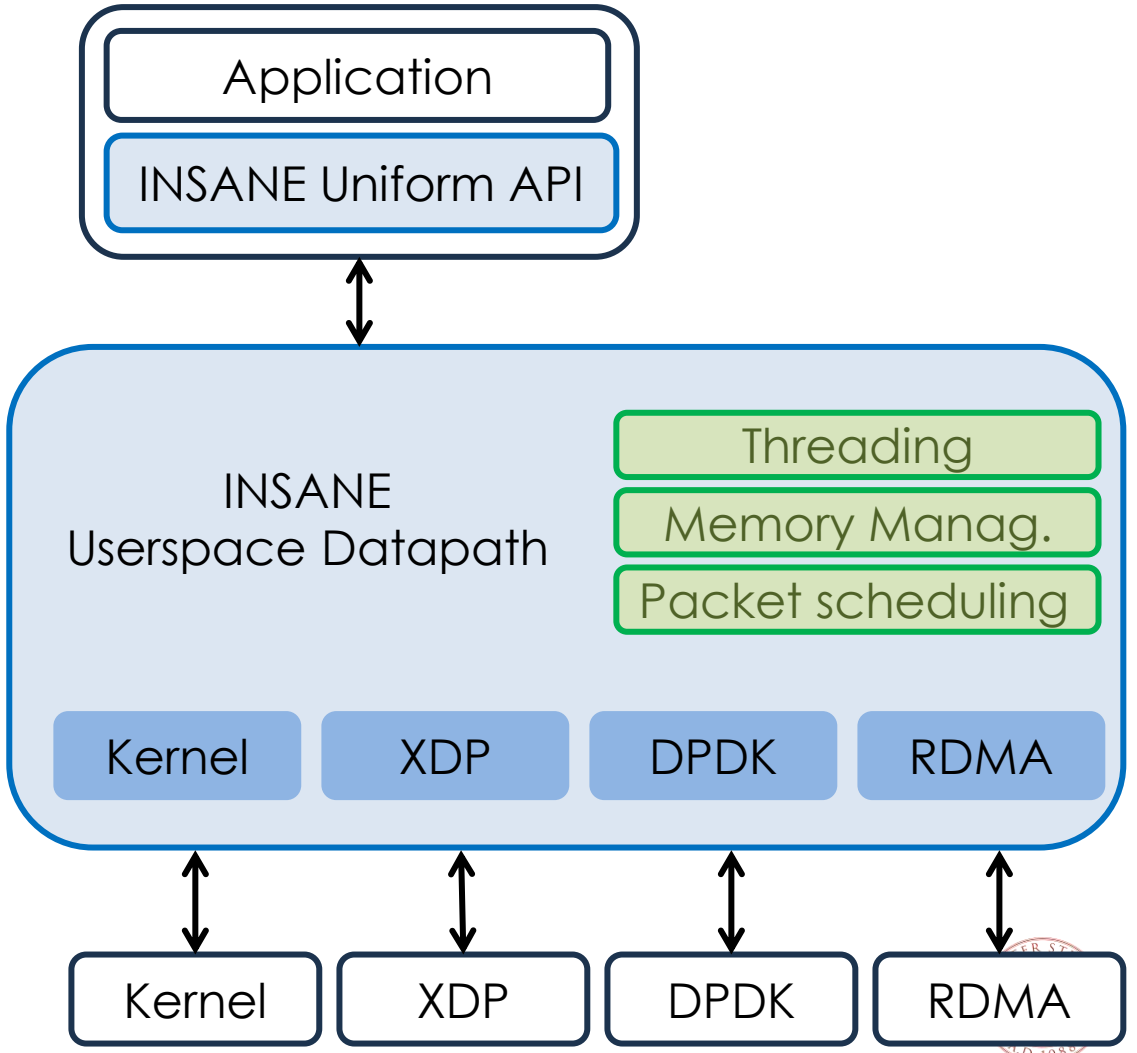
INSANE: Integrated aNd Selective Acceleration for the Network Edge

[ACM Middleware'23]

INSANE offers a portable, accelerated datapath by providing:

1. A **simple custom API**: minimal primitives designed to support heterogeneous network stacks.

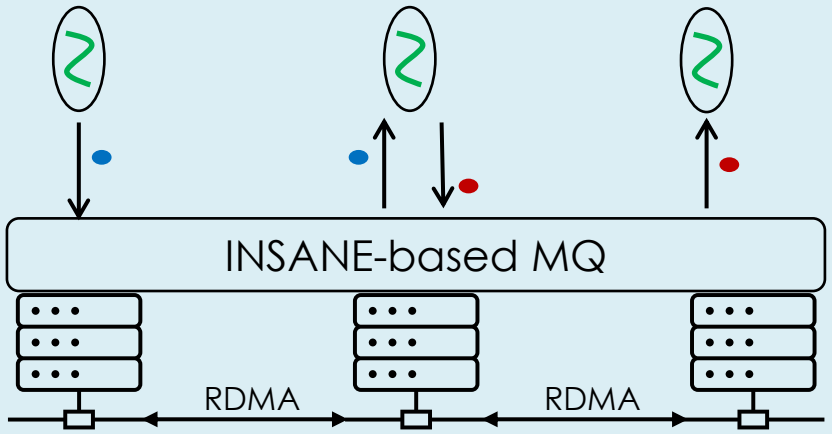
2. A **userspace module**: provides µs-scale OS features in a plugin-based, centralized userspace datapath.



Applications of INSANE (1/3) : Cloud Message Queue

Use Case: Serverless

FaaS platform for the transparent composition of serverless functions



Serverless computing. Serverless platforms already use a messaging system (message queue, MQ) to exchange control and data messages among functions.

An INSANE-based MQ can exploit modern network hardware to enable the same zero-copy interaction that characterizes local shared-memory communication.

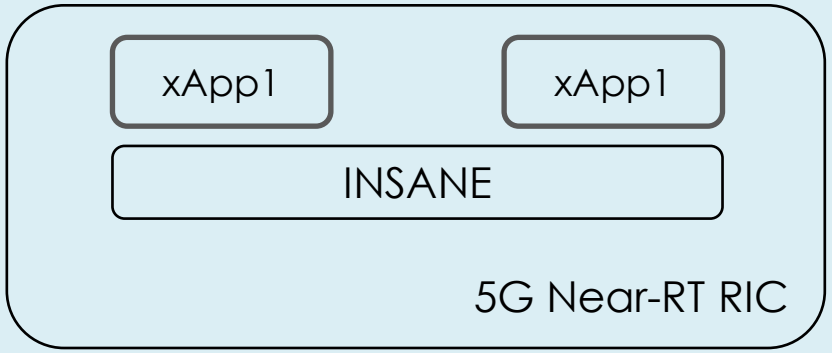
Applications of INSANE (2/3) : The 5G Infrastructure

5G Infrastructure. In the control path of 5G, there are apps (xApps) that need fast communication with other apps to support decisions such as scheduling, interference mitigation, and mobility management.

INSANE enables *differentiated service* by allocating the highest-performance backend (e.g., DPDK) to premium or time-critical users when resources permit.

Use Case: 5G integration

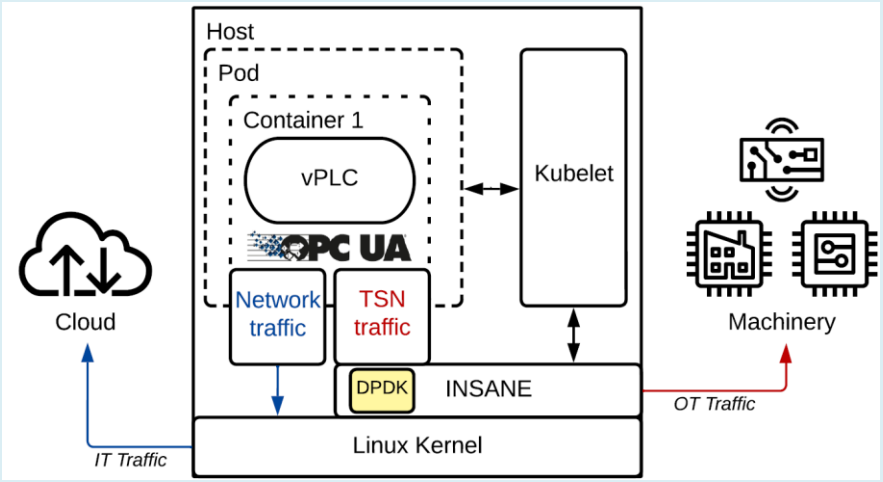
Support communications of xApps in the near-RT RIC within the O-RAN architecture



Applications of INSANE (3/3): The Industry 4.0 Ecosystem

Use Case: vPLC

Coexistence of industrial applications with heterogeneous requirements (real time, best effort).



IT/OT integration. Running industrial control (e.g., virtual PLCs, or vPLCs) on cloud-style IT platforms while preserving real-time behavior.

INSANE provides deterministic, kernel-bypassing communication so vPLCs in containers still meet strict timing guarantees.

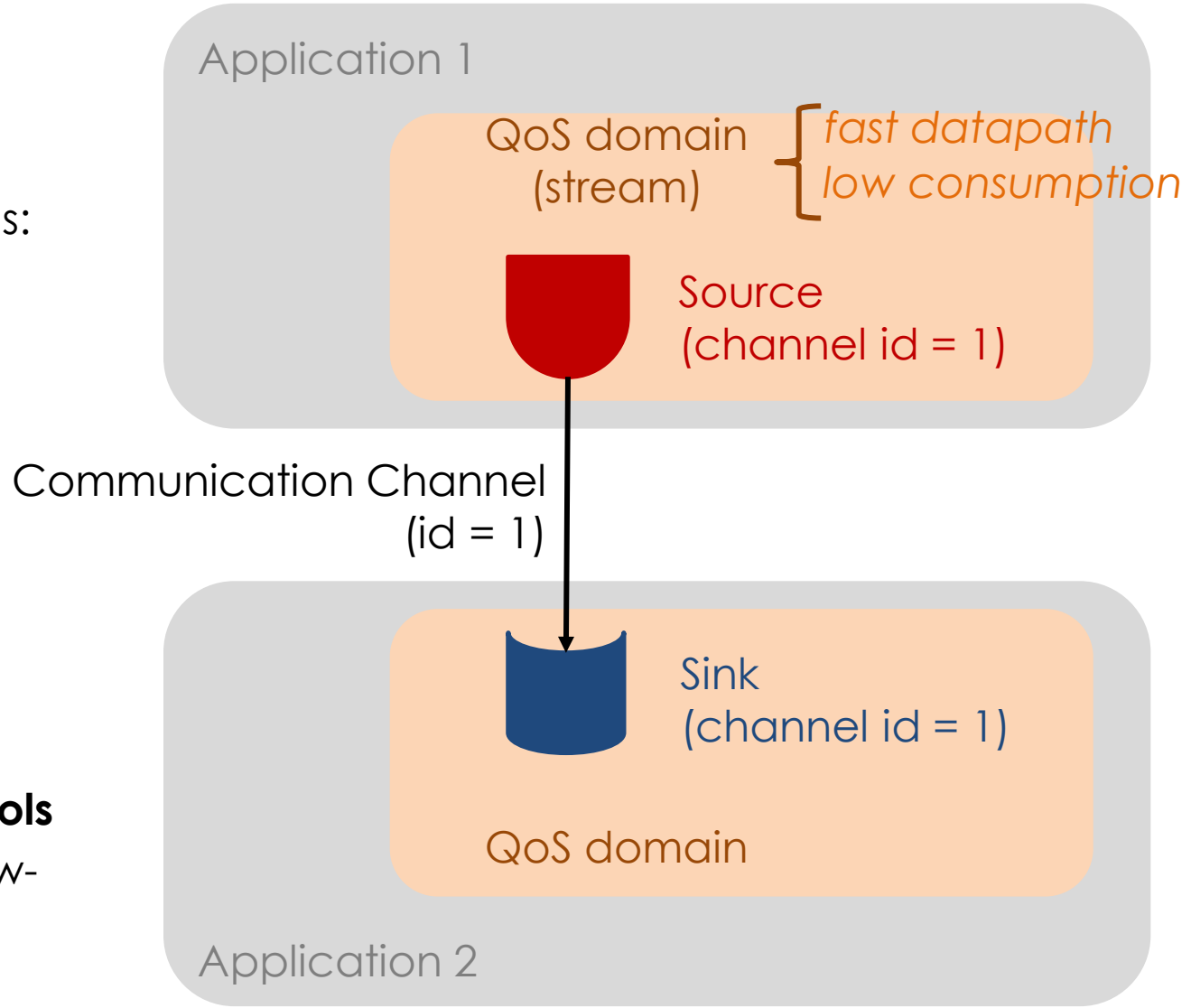
The INSANE Portable API: design goals

The INSANE API answers three key design goals:

1. **Ease of use**
   Few simple concepts: *communication channels*, *streams*, *sources*, and *sinks*.

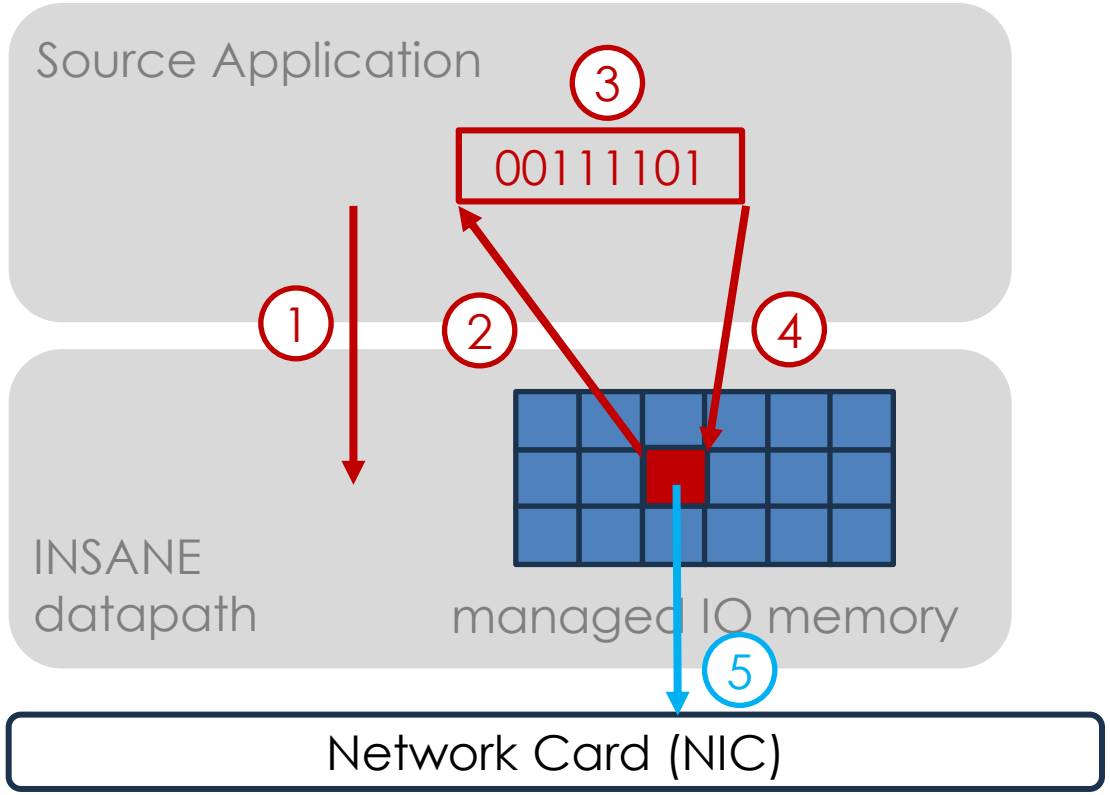
2. **General-purpose expressiveness**
   Flexibility to implement higher-level and domain-specific abstractions.

3. **Agnostic to the underlying network protocols**
   Preserve the zero-copy semantic without low-level details.



The INSANE Portable API: Zero-Copy transfers

The INSANE API is **asynchronous** and lets apps access a DMA-capable heap for **zero-copy I/O**.



Clear **memory ownership semantic**: memory is allocated by INSANE, and applications borrow *memory slots* from it.

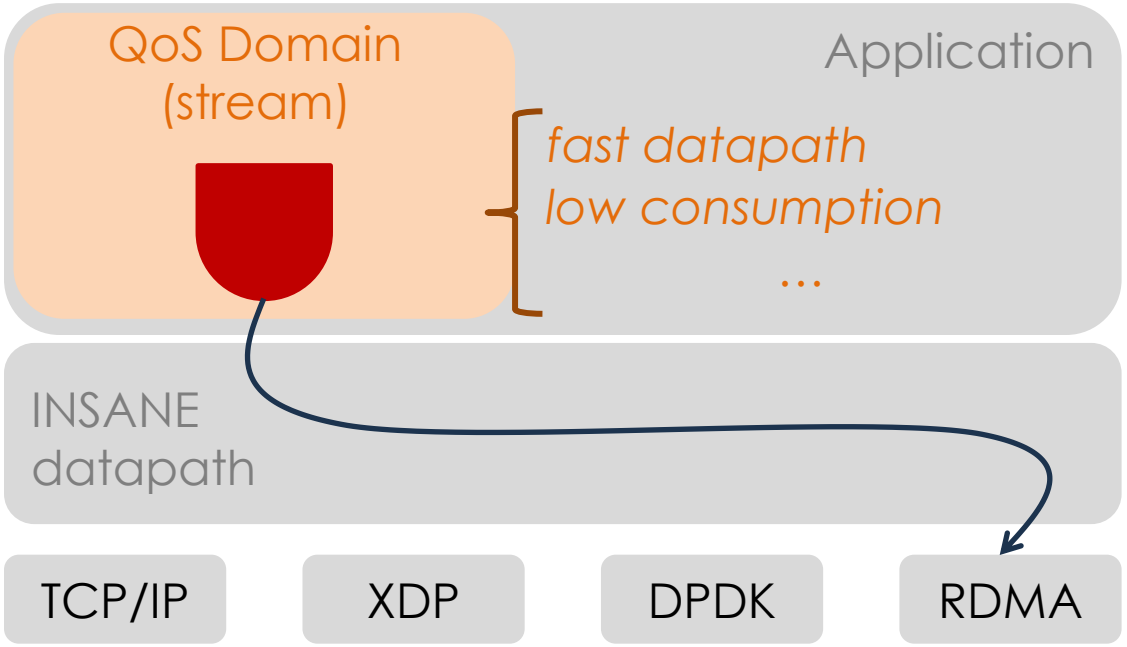
The INSANE OS module performs the **actual network operations** by selecting a specific network technology.

The INSANE Portable API: QoS-based technology selection

To enhance portability, INSANE moves from the user to the middleware the choice of **which network technology** to associate to communication channels.

Users can provide hints via **QoS policies,** e.g.:

- *Reliability: unreliable, reliable*
- *Datapath Acceleration: fast, standard.*
- *Resource Consumption: low, standard.*



INSANE combines **QoS policies** and the **local availability** to decide which network technology to associate to channels of the same *QoS domain*.

INSANE: The API

Currently available in

- C
- Python

```c
/* Open and close a session */
int init_session();
int close_session();

/* Stream */
stream_t create_stream(options_t opts);
void close_stream(stream_t stream);

/* Source APIs */
source_t create_source(stream_t stream, int channel);
void close_source(source_t source);
buffer_t get_buffer(source_t src, size_t size, int flags);
int emit_data(source_t src, buffer_t buffer);
int check_emit_outcome(source_t source, int id);

/* Sink APIs */
sink_t create_sink(stream_t stream, int channel, data_cb cb);
void close_sink(sink_t sink);
int data_available(sink_t sink, int flags);
buffer_t consume_data(sink_t sink, int flags);
void release_buffer(sink_t sink, buffer_t buffer);
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

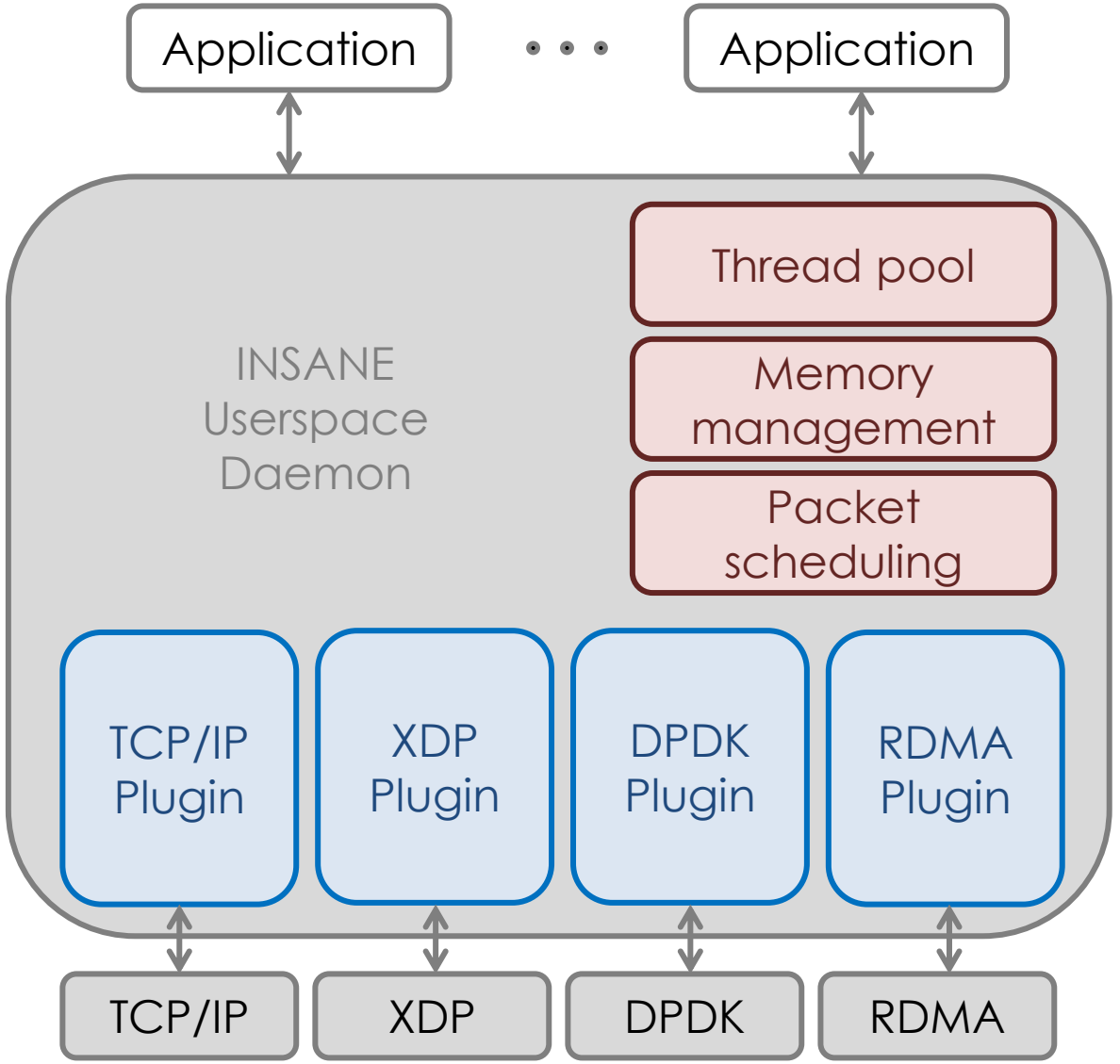
The INSANE Userspace Datapath

The userspace module centralizes OS features to support a **general-purpose datapath**:

1. *memory management* for zero-copy transfers
2. *packet scheduler* for traffic shaping
3. a *pool of threads* for network operations

A set of *datapath plugins* implements the necessary network operations for each supported acceleration technology.

This design enables **portability** and **transparency** of the accelerated datapath.



The INSANE Userspace Datapath

The userspace datapath follows a "*sidecar*" approach:
(Snap [SOSP'19] – TAS [EuroSys'19])

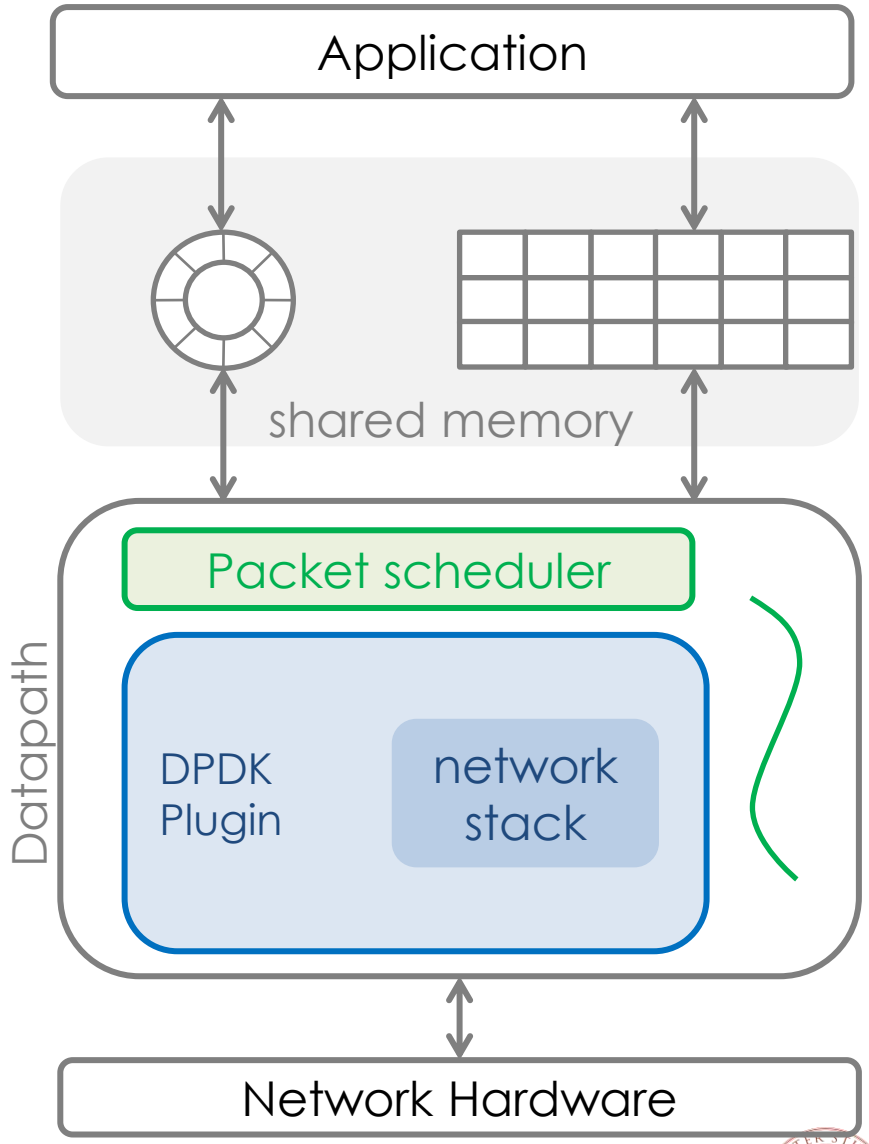
Datapath functionalities are **centralized** into a separate process than the application.

Advantages:

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

Disadvantages:

- IPC between apps and the datapath process



The INSANE network plugins

Plugins bridge the gap between INSANE's high-level abstractions and the concrete capabilities of each accelerated networking technology.

- Specialize INSANE's agnostic datapath for specific network backends. This enables **modularity and extensibility**: new backends can be easily added.

- Provide a **userspace network stack** when required by the backend (XDP, DPDK)

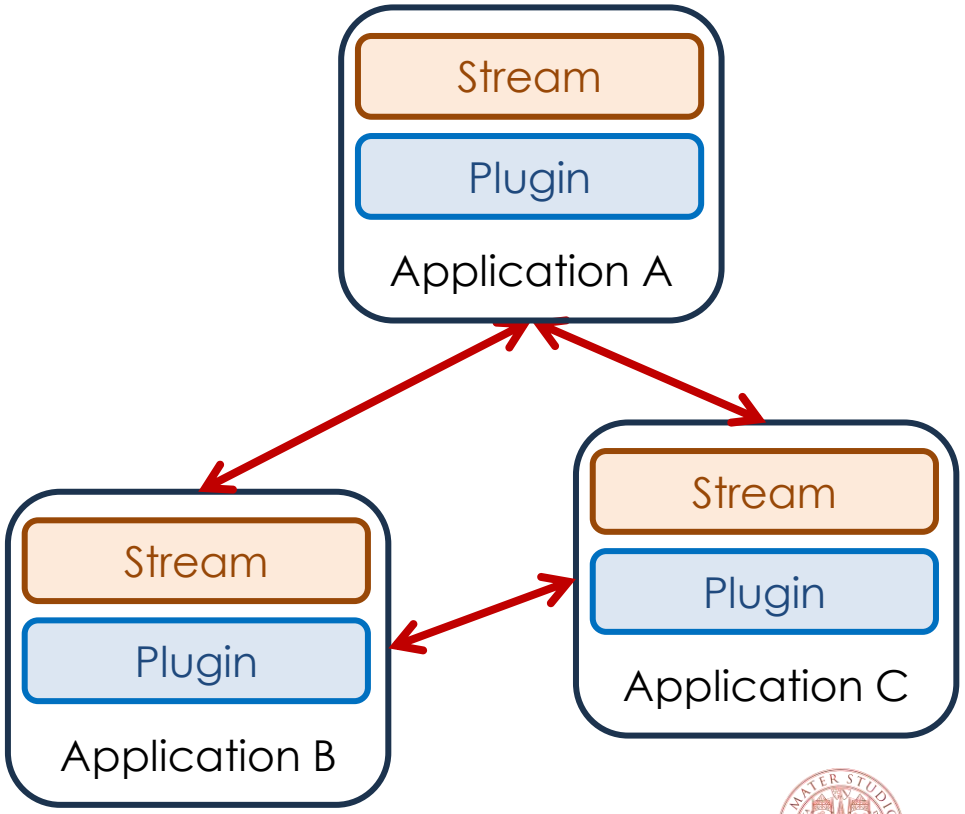
- Run in the same address space as the INSANE runtime, enabling direct access to application buffers (**zero-copy I/O**).

The INSANE network plugins: *overlay networks*

Plugins implement **streams** as **dynamic overlay networks**, handling setup, lifecycle, and failures.

- **On-demand overlays**. A stream triggers creation of an overlay only when communication starts.
- **Dynamic participation**. Only peers with active applications for that stream join; others can join later.
- **Shared fabric**: one overlay per stream, shared across channels.

<u>Communication is a multicast over the overlay network</u>

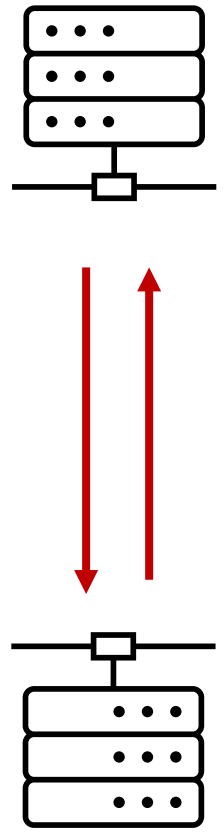


Performance evaluation

1) Evaluate the **overhead** INSANE introduces on the datapath
   Compared to:
   - Raw Acceleration Technologies
   - Demikernel [SOSP '21]

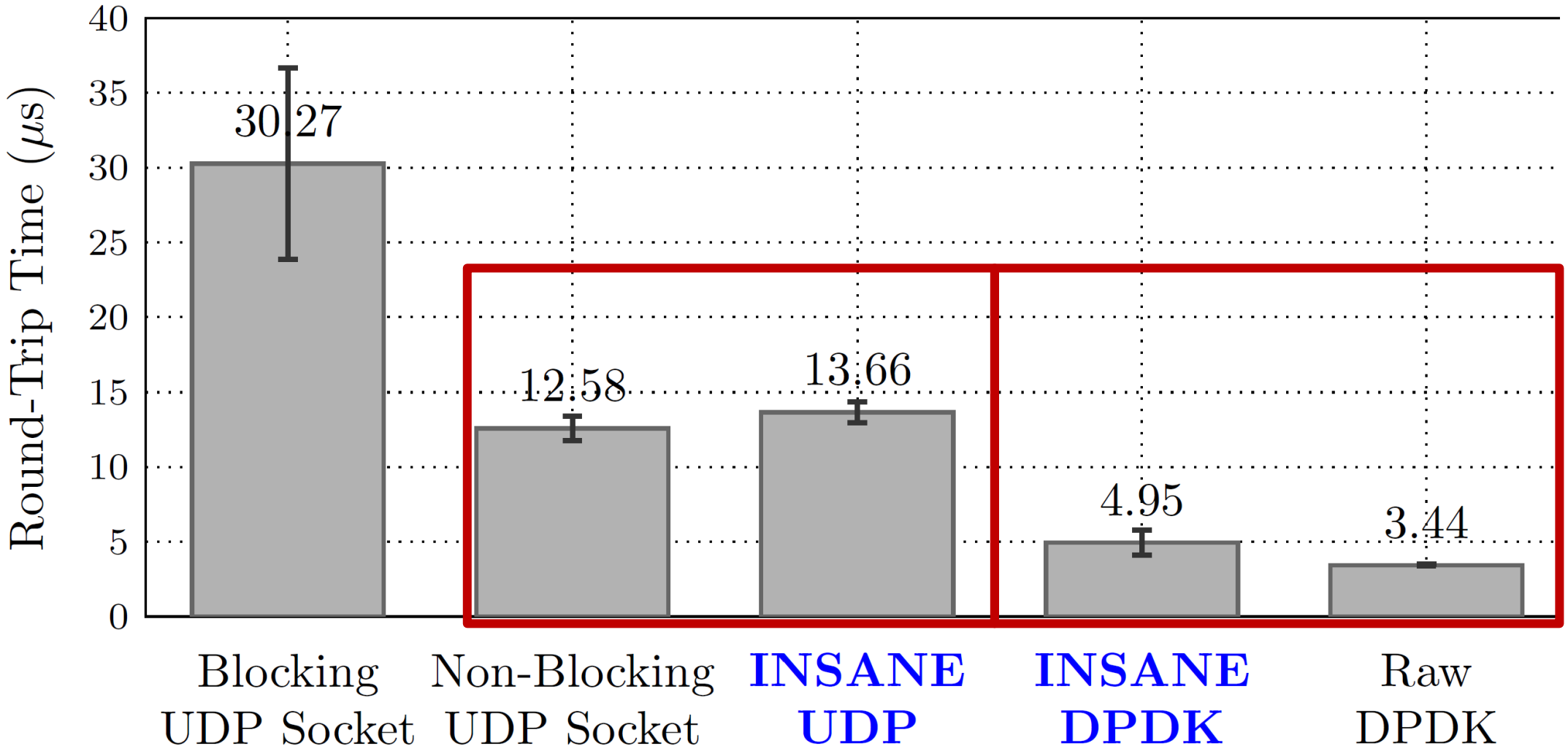
Latency test: a ping-pong application to measure the RTT of messages through various data paths.

2) Assess the ease of use of the INSANE API
   - Implement a typical image streaming service

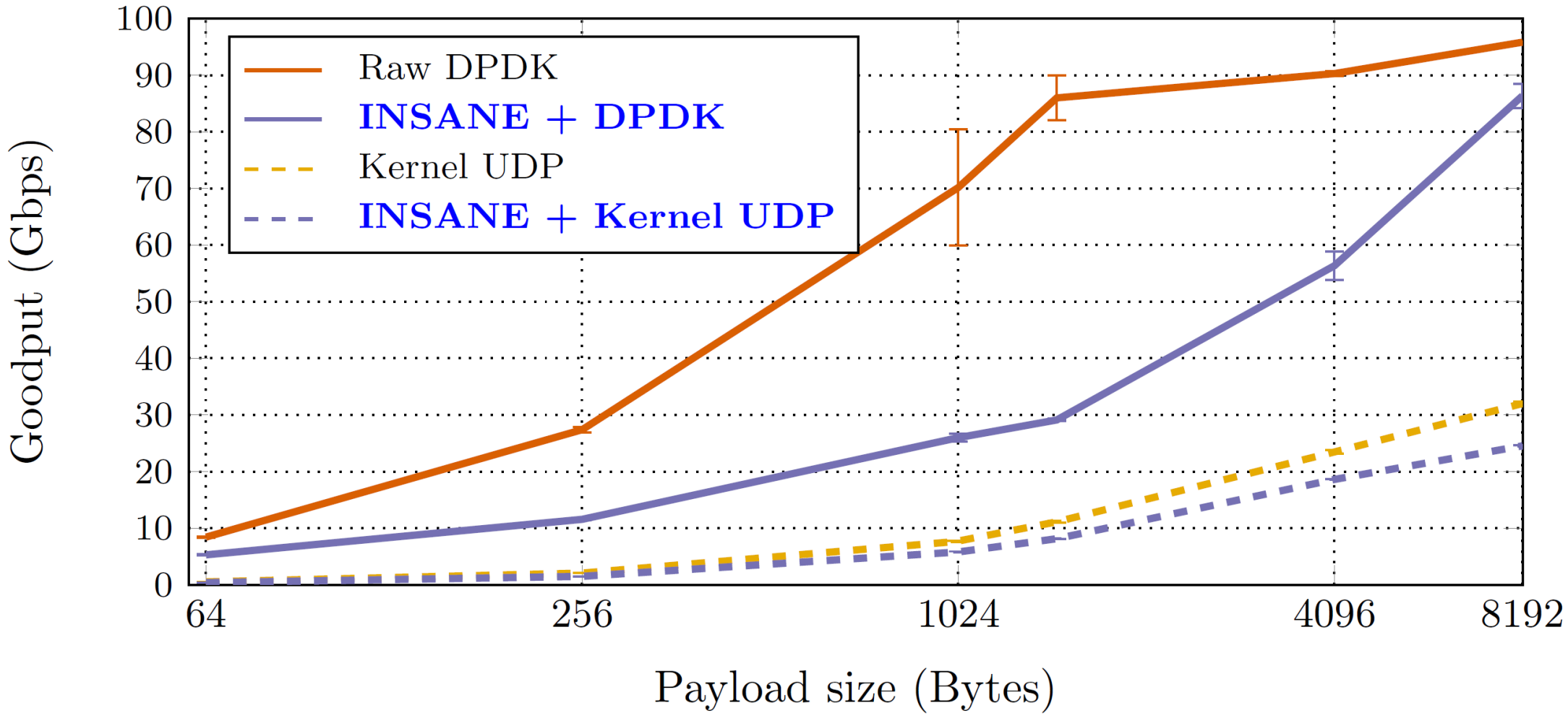


CPU: 18-core Intel i9 RAM: 64GB
Network: Mellanox DX-6 100Gbps

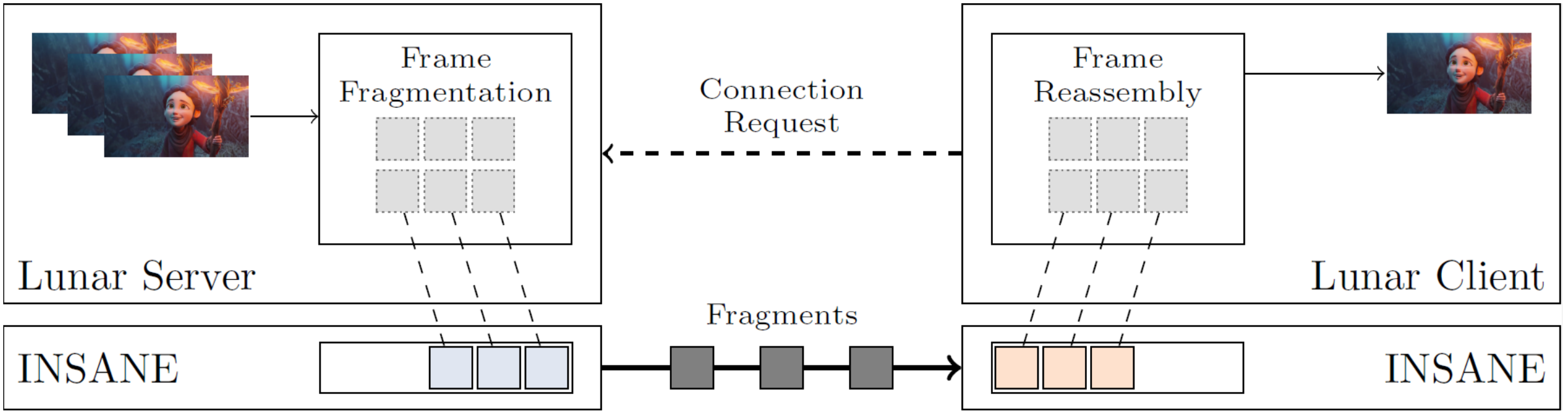
Performance evaluation: Latency test for 64B payload size



Performance evaluation: Throughput test



Performance evaluation: The LUNAR streaming framework

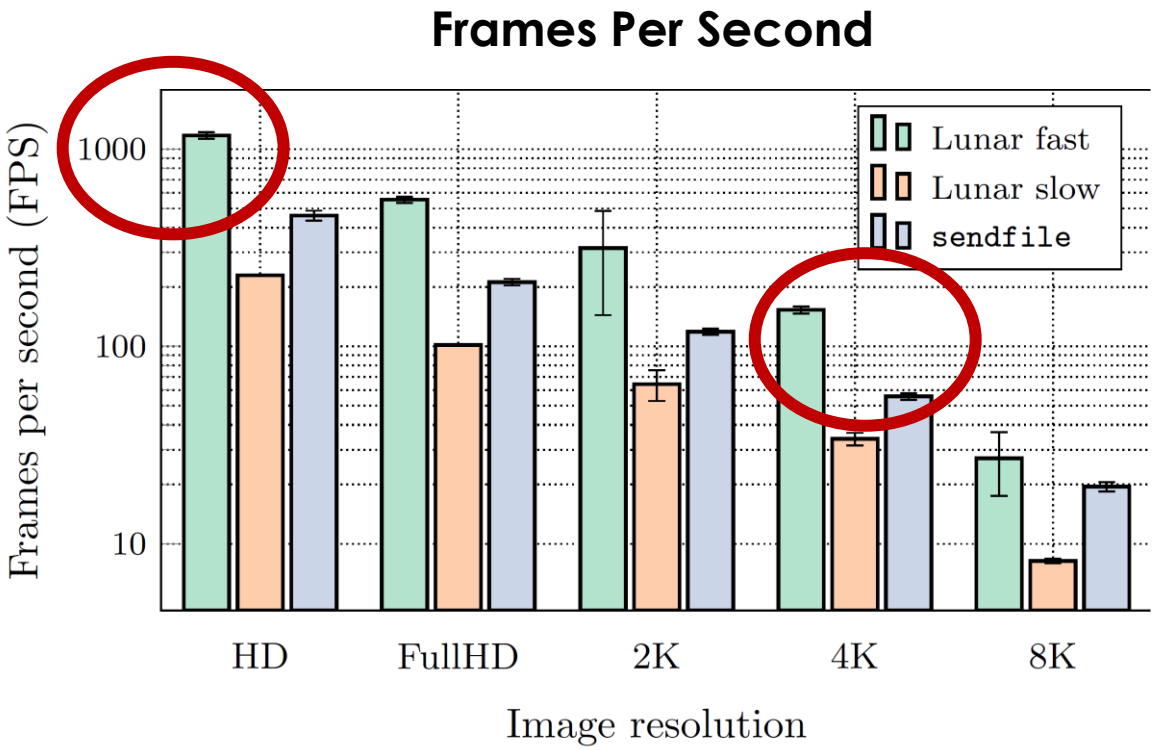


The implementation of this framework was easy: only 200 Lines of Code

Benchmark: number of **frames per second (FPS)** received by the client

Baseline: zero-copy in-kernel transmission using the **`sendfile()`** primitive.

Performance evaluation: The LUNAR streaming framework



Excellent for applications such as **Tactile Internet** or near **real-time image inference** (e.g., autonomous vehicles, industrial automation).

Resolution	HD	Full HD	2K	4K	8K
Size (MB)	2.76	6.22	11.6	24.88	99.53

Future work: Coordinating Hardware Offloading

Edge cloud relies on the same abstractions of the cloud (containers, VMs)

SW/HW codesign will be crucial to offer *cloudification* support to fast I/O:

- enable scalable and secure virtualization and sharing of acceleration devices
- improve the flexibility and programmability of hardware devices

Infrastructure Programmer Development Kit (IPDK)

ipdk Infrastructure Programmer Development Kit

Data Processing Units (DPU)

e.g., NVIDIA BlueField