Advanced Computer Networks

CS740

SmartNICs

https://pages.cs.wisc.edu/~mgliu/CS740/F25/index.html

Ming Liu
mgliu@cs.wisc.edu

Outline

- Last lecture
  - Programmable Switch
- Today
  - SmartNICs
- Announcements
  - Lab2 due 11/05/2025 11:59 PM
  - Midterm report due 11/04/2025 11:59 PM

What is a SmartNIC?

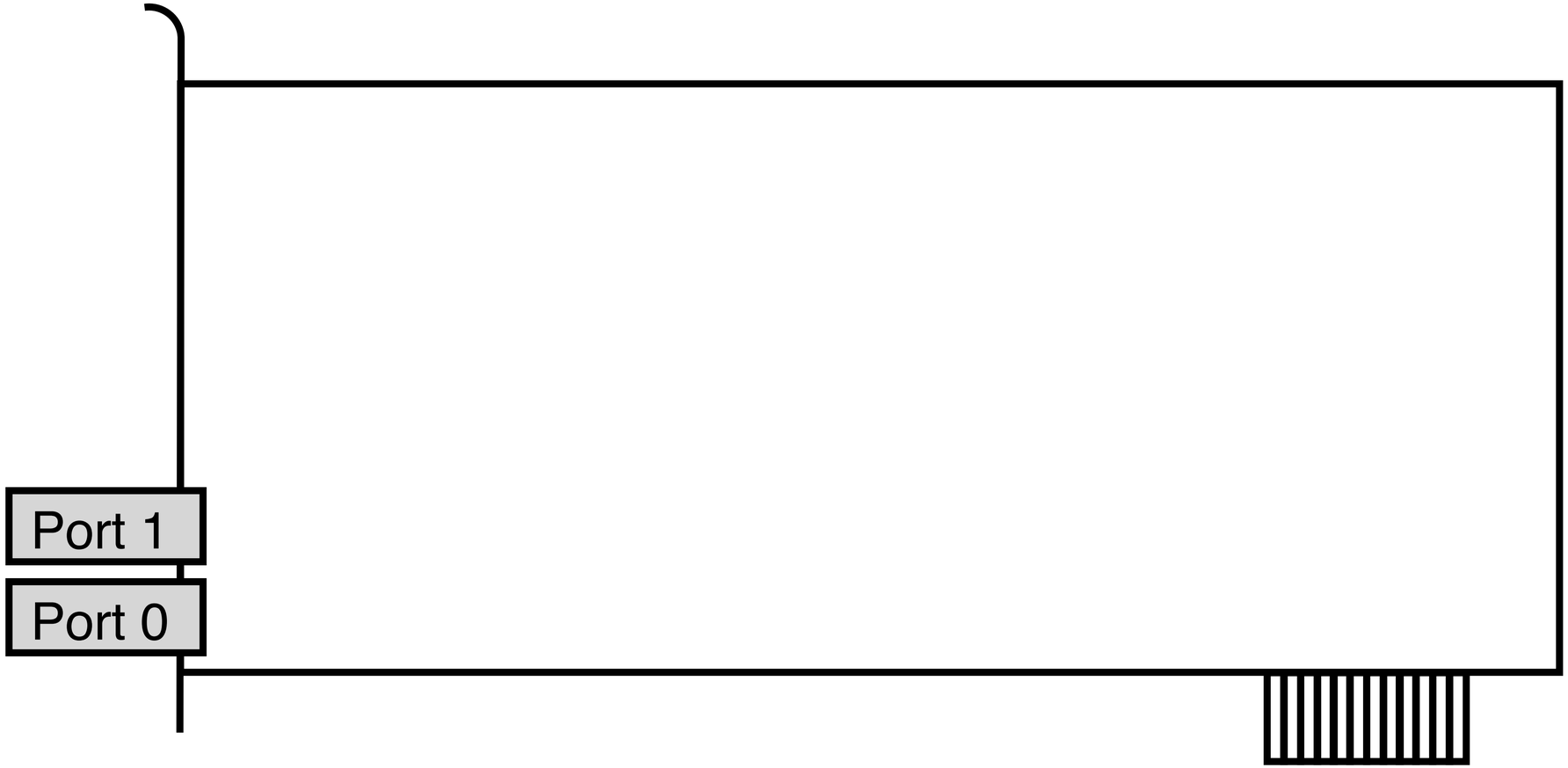
What is a SmartNIC?

Wait, what is a NIC?

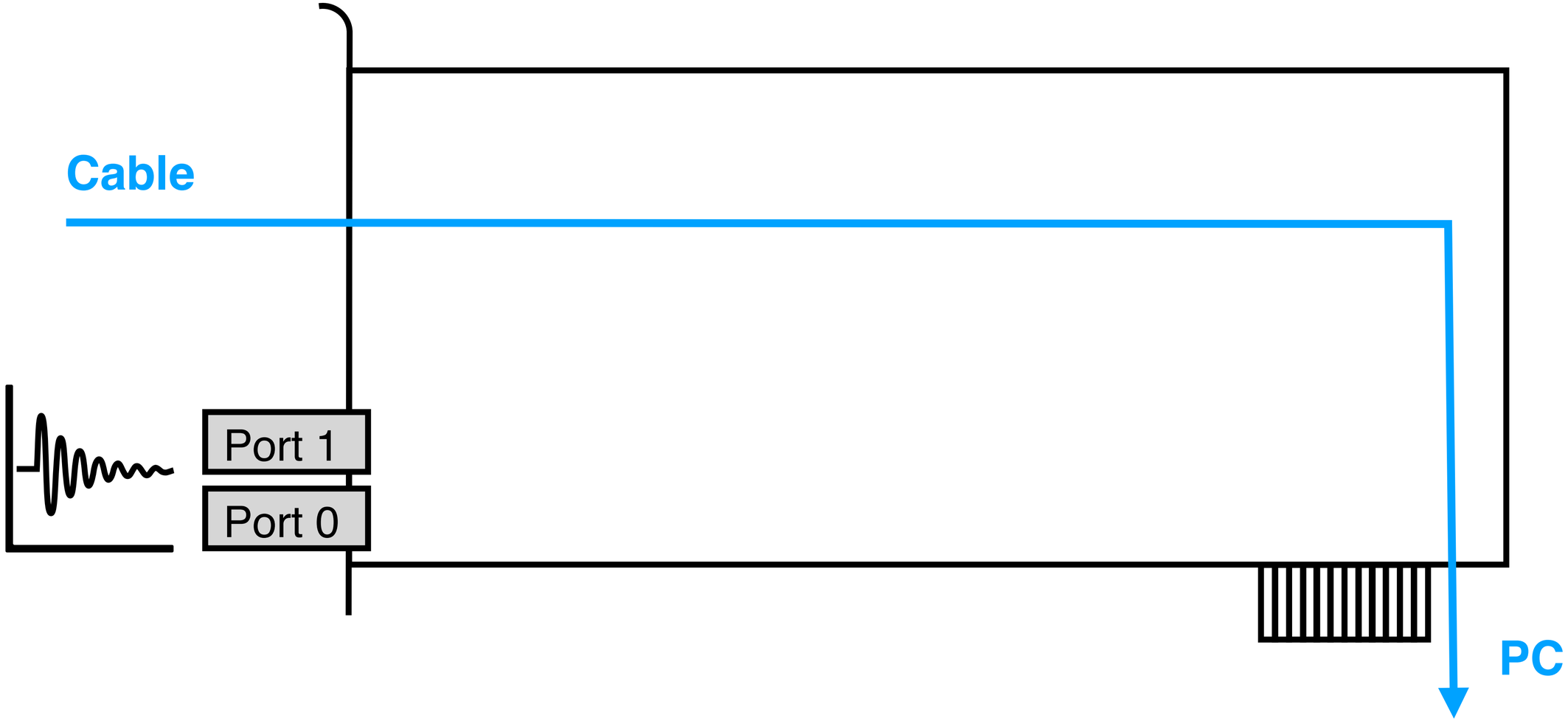
NIC (Network Interface Card)
= A specialized computer hardware that connects your system to a computer network.

What does a NIC do?

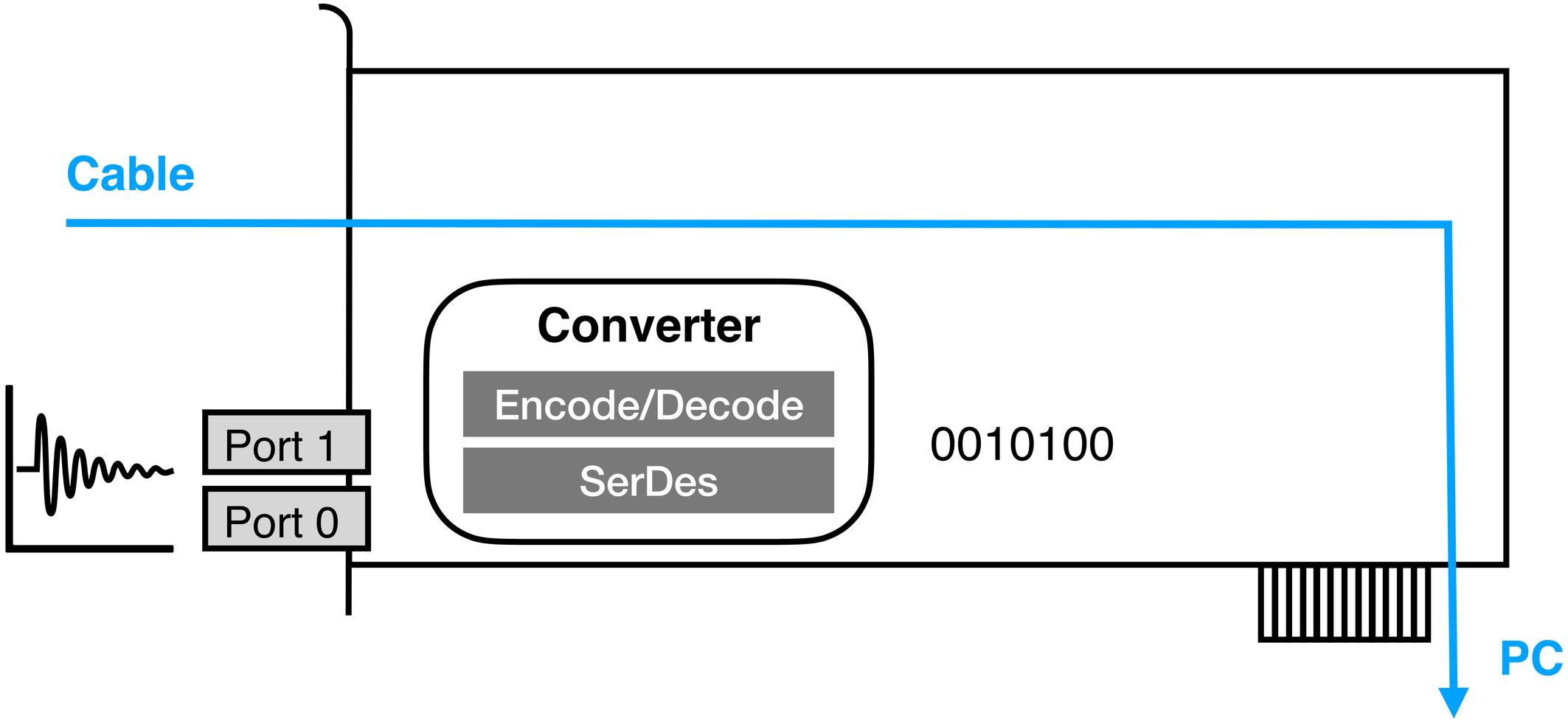
What does a NIC do?



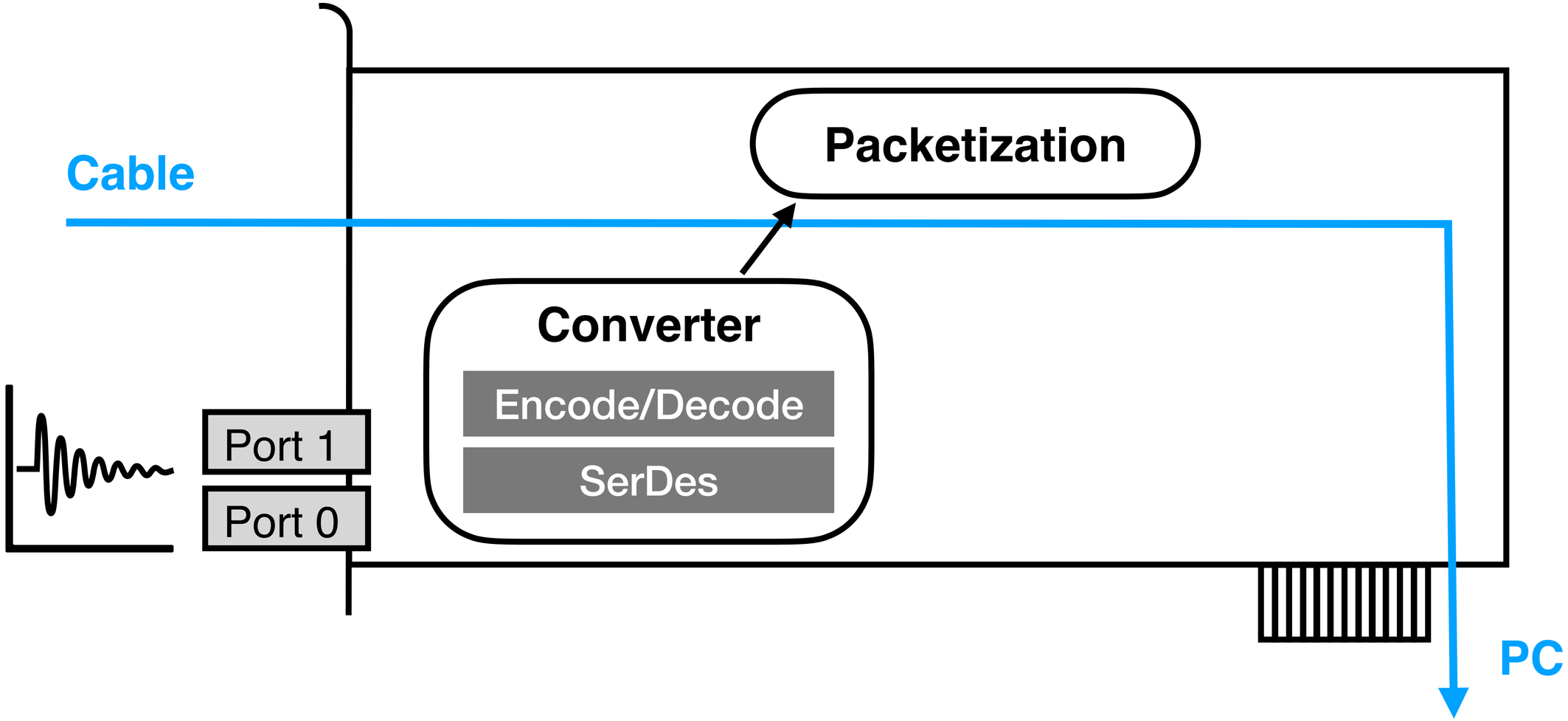
What does a NIC do?



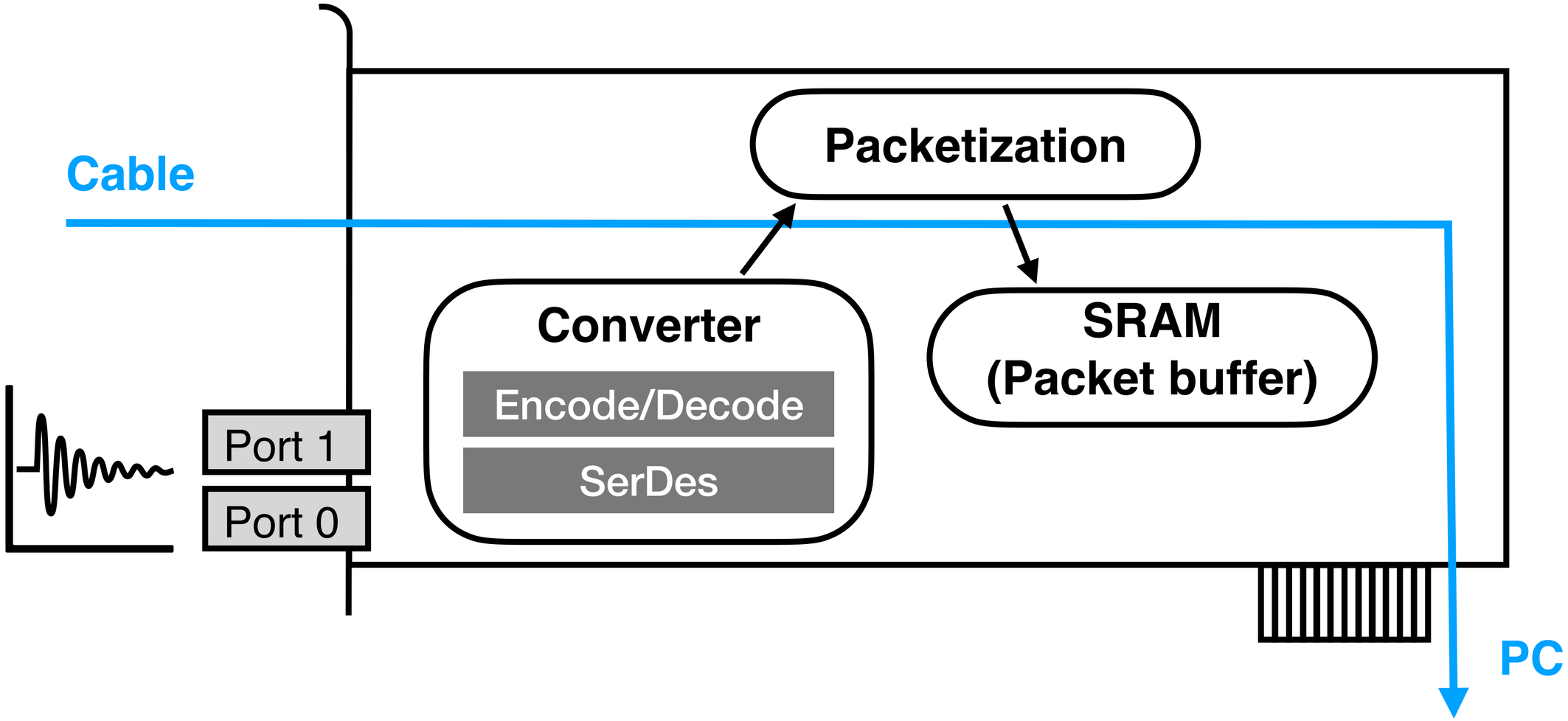
What does a NIC do?



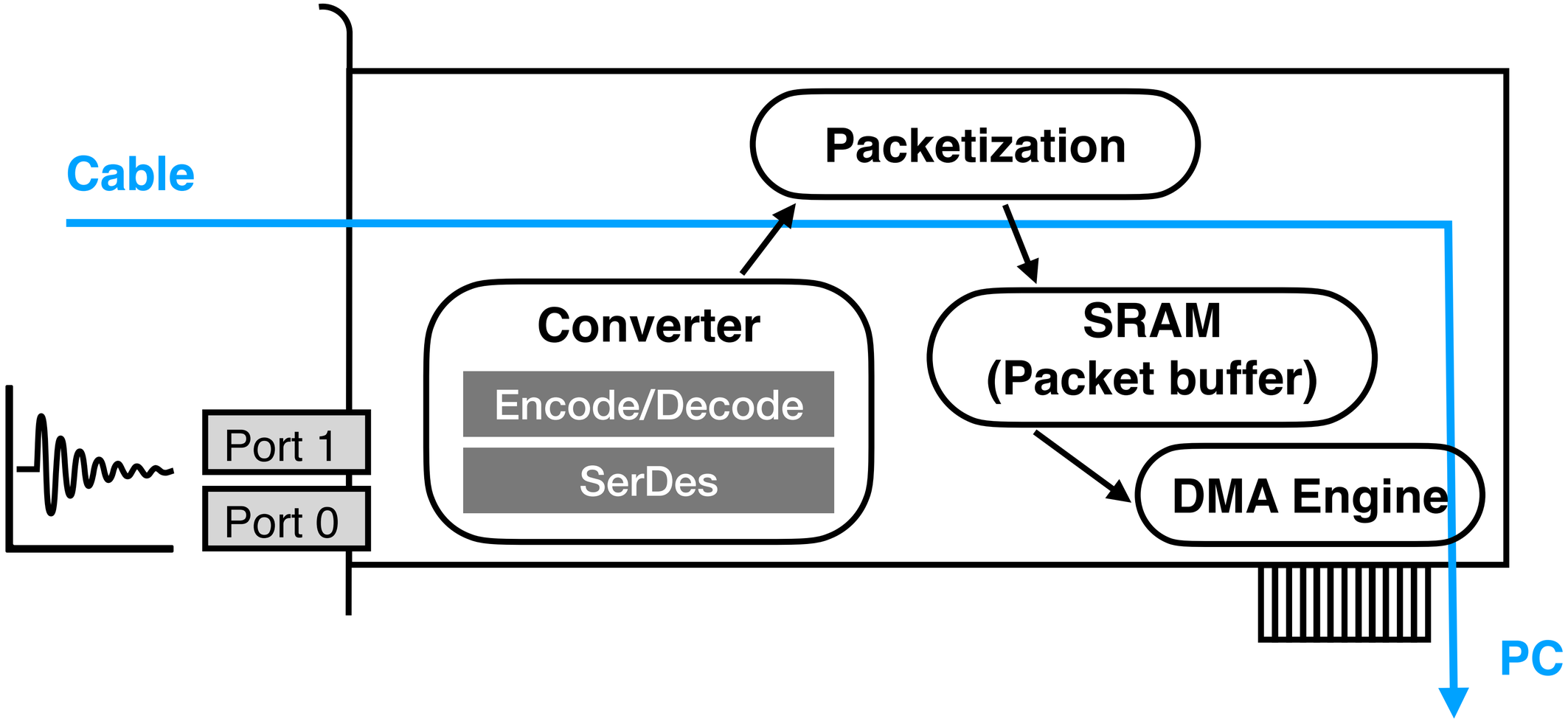
What does a NIC do?



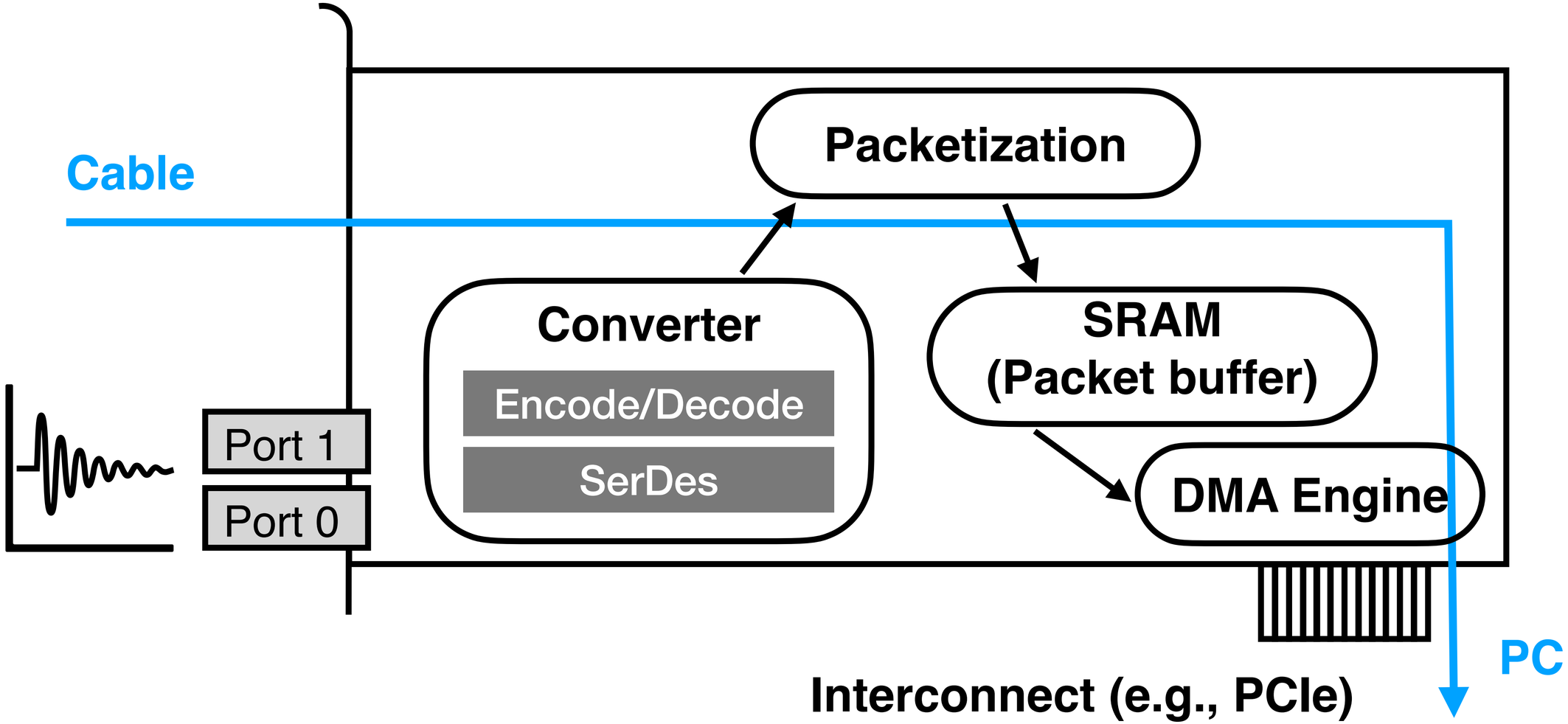
What does a NIC do?



What does a NIC do?



What does a NIC do?



What is a SmartNIC?

What is a SmartNIC?

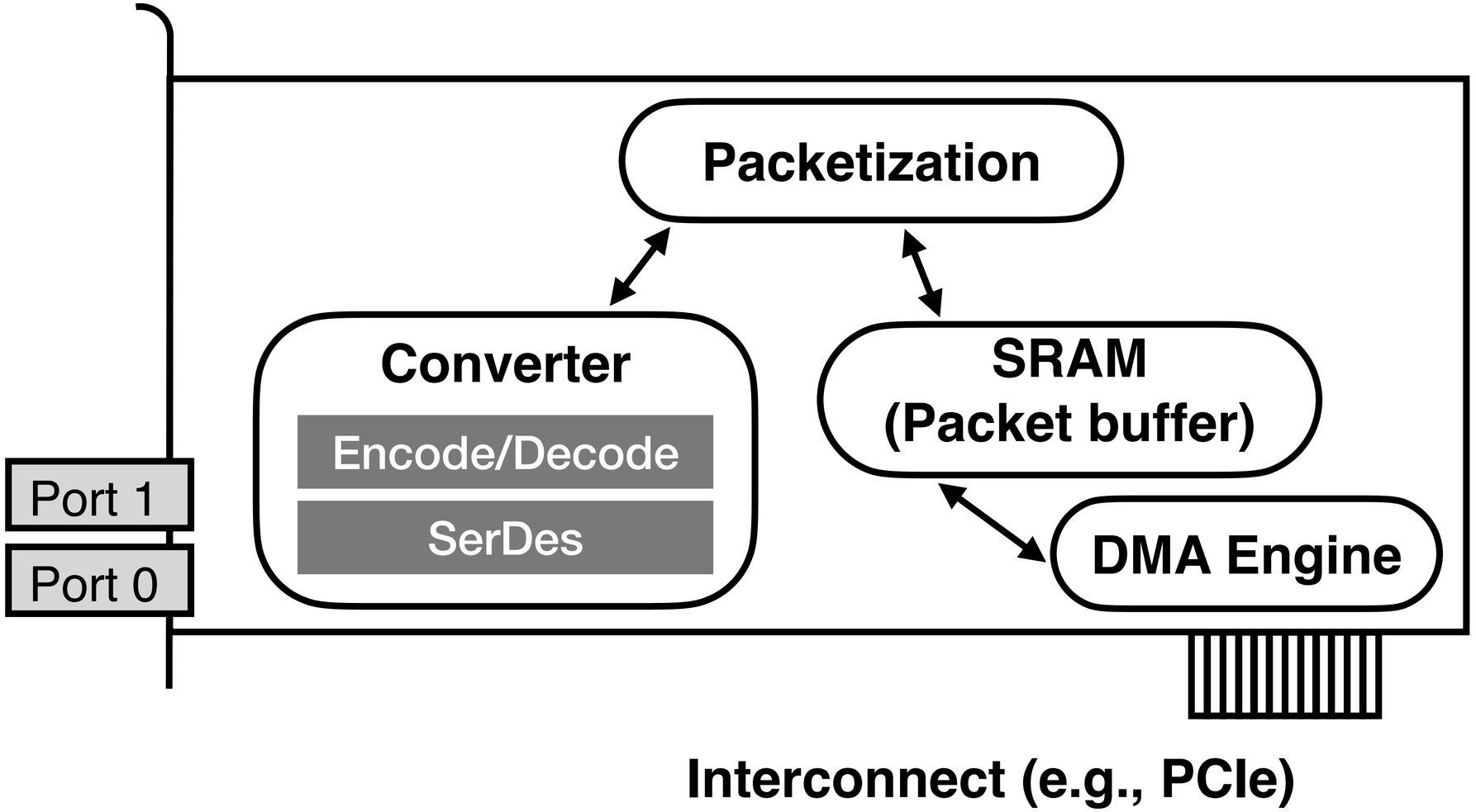
DPU (Data processing unit)

IPU (Infrastructure processing unit)

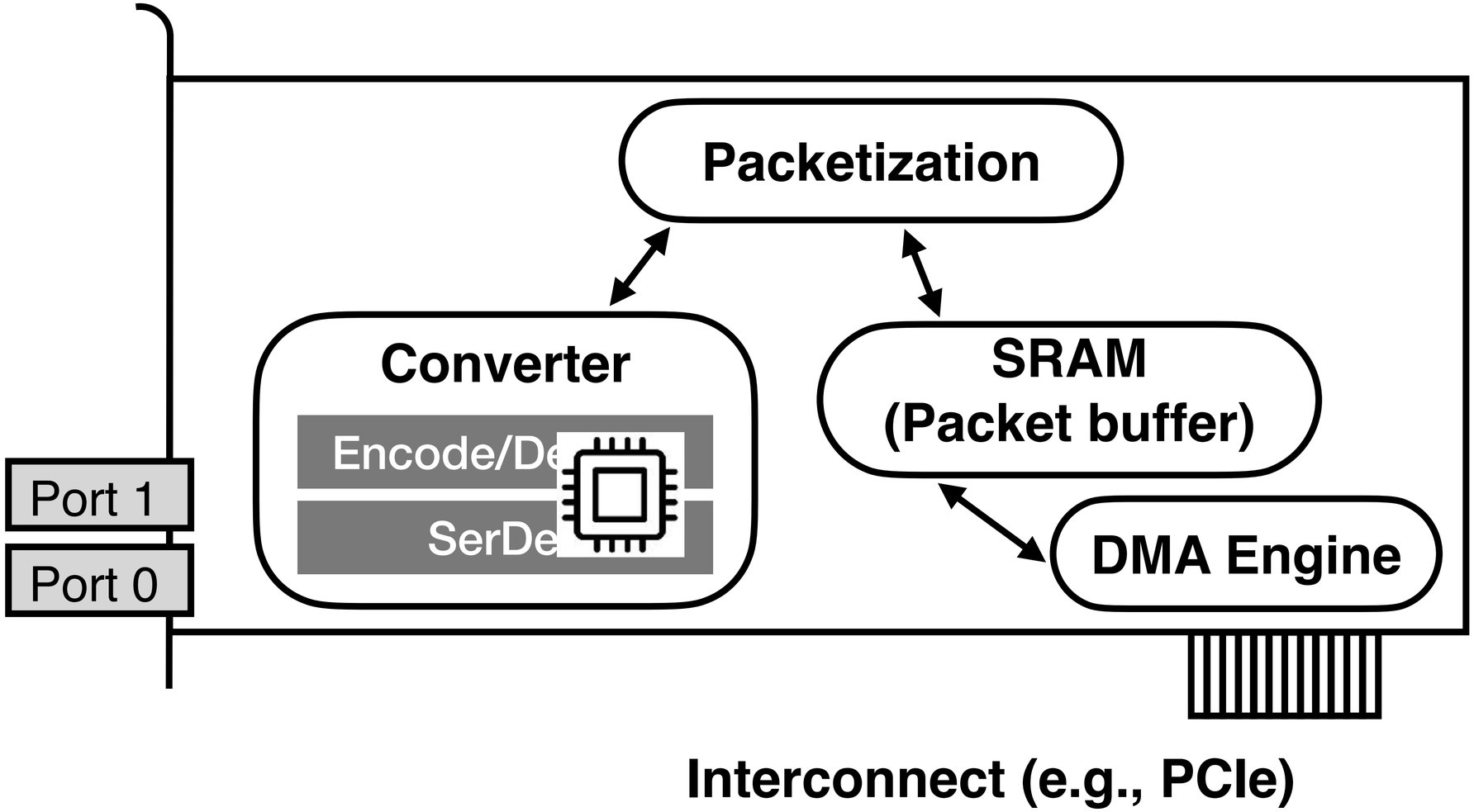
What is a SmartNIC?

SmartNIC=NIC+Reconfigurable Computing

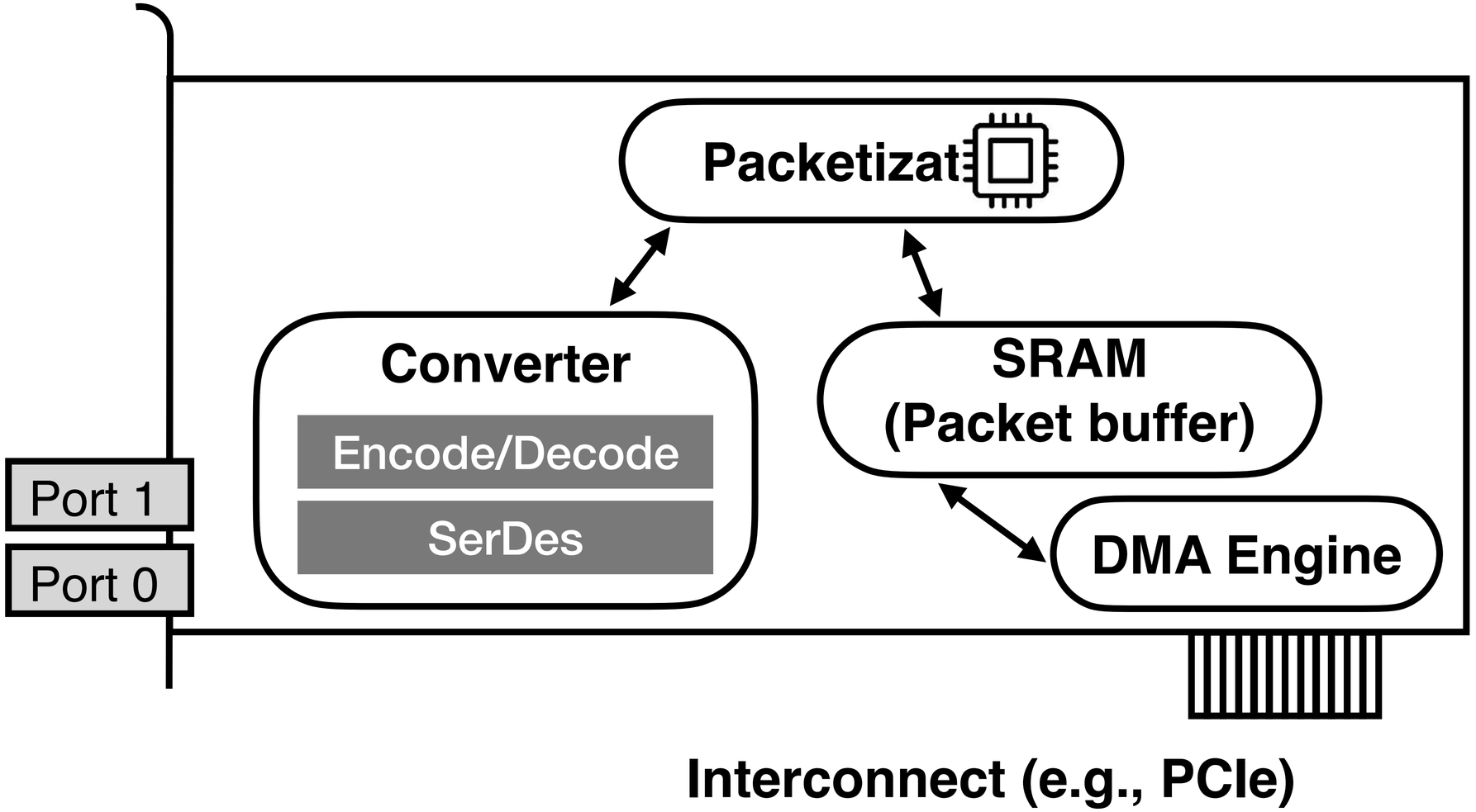
How to make a NIC “Smart”?



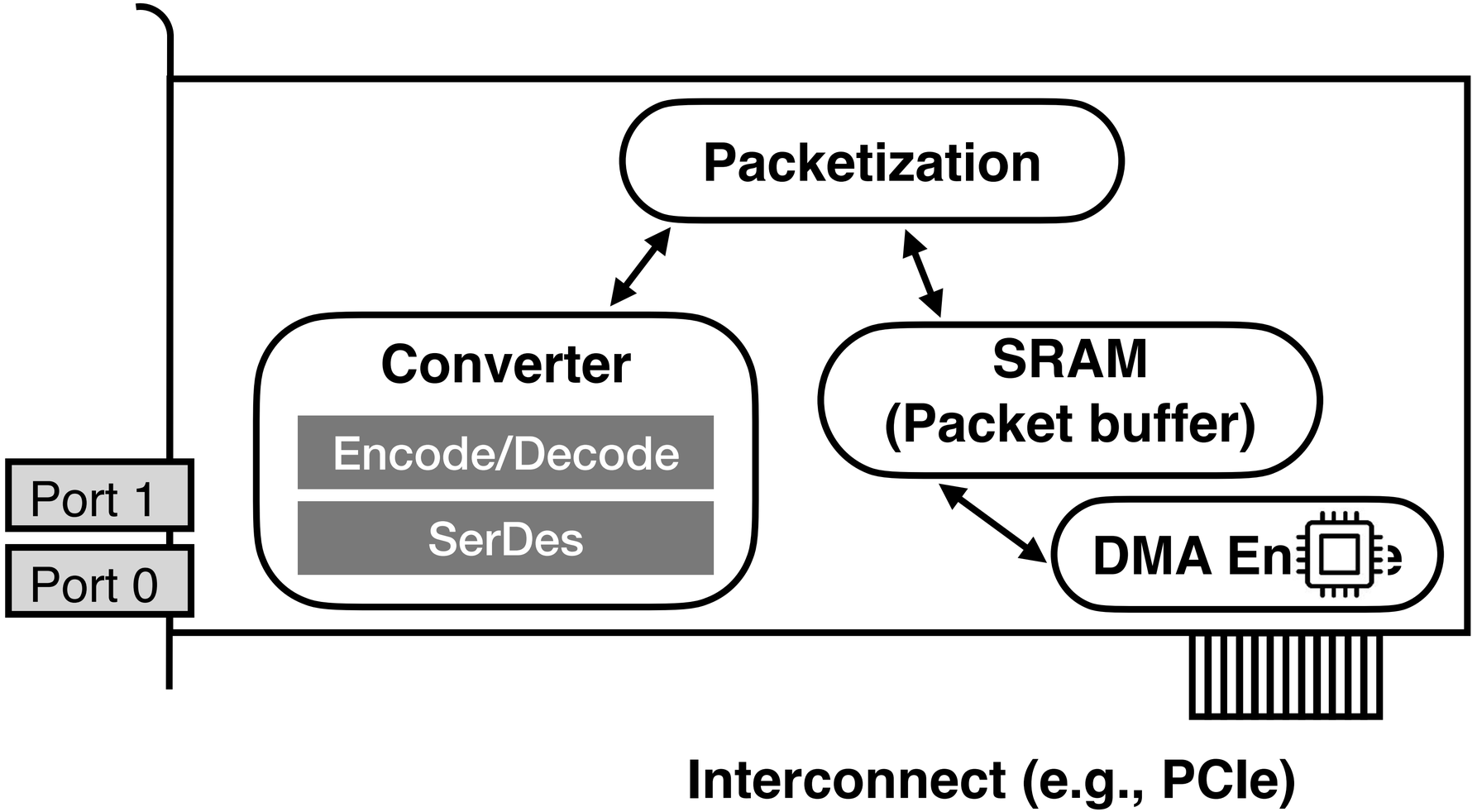
How to make a NIC “Smart”?



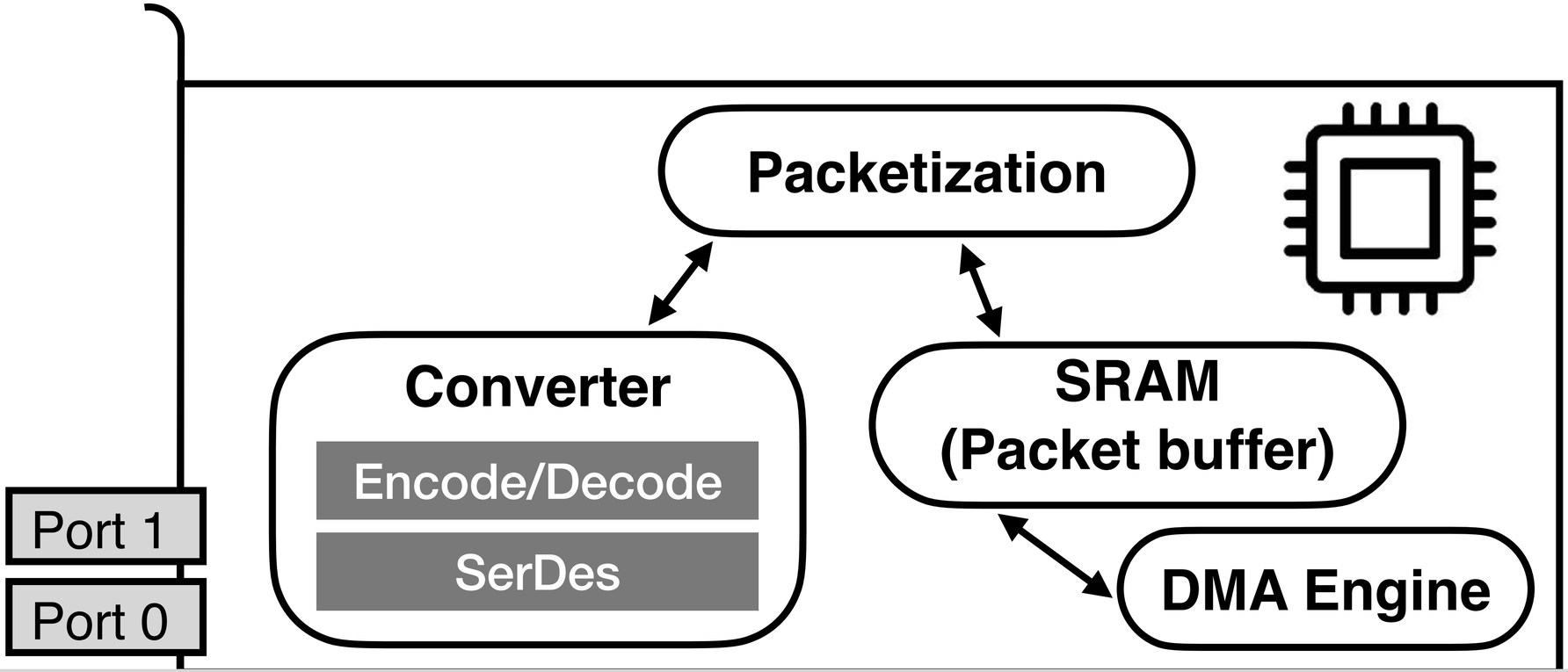
How to make a NIC “Smart”?



How to make a NIC “Smart”?



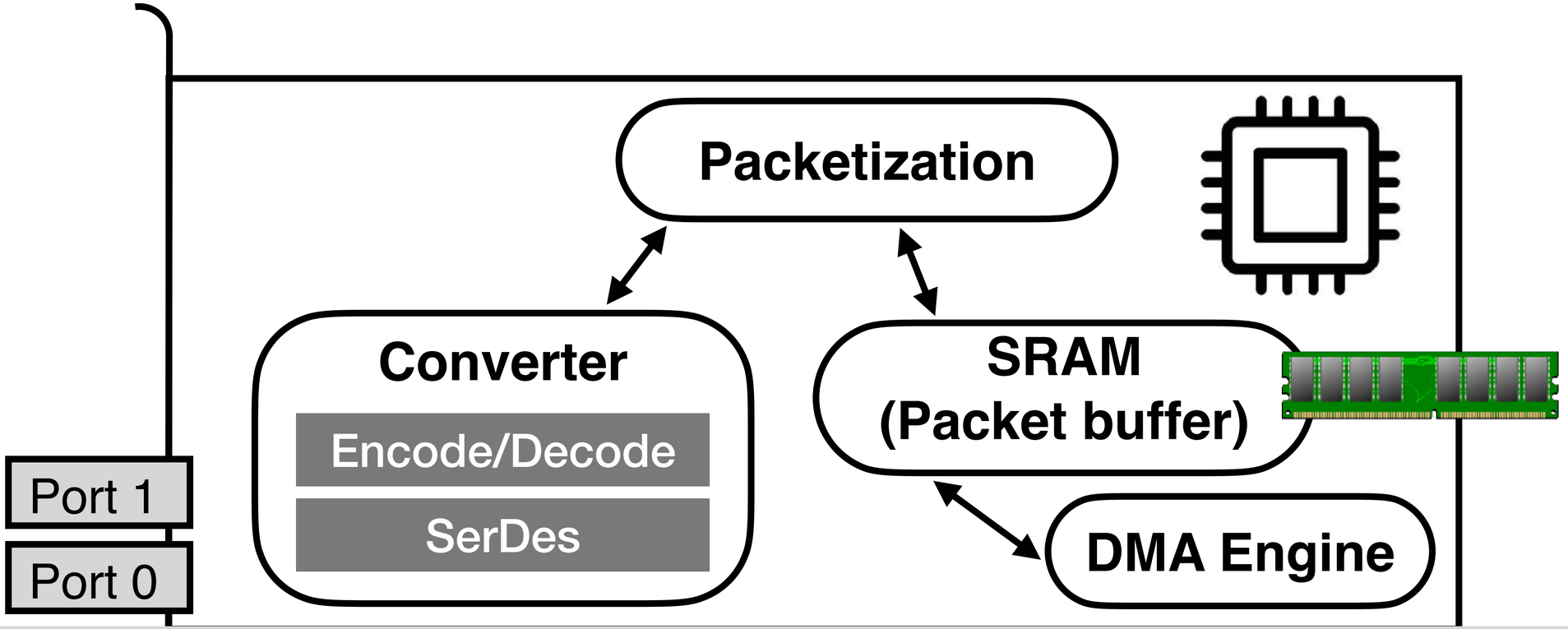
How to make a NIC “Smart”?



#1: Add compute

- Computing core
- Architectural position

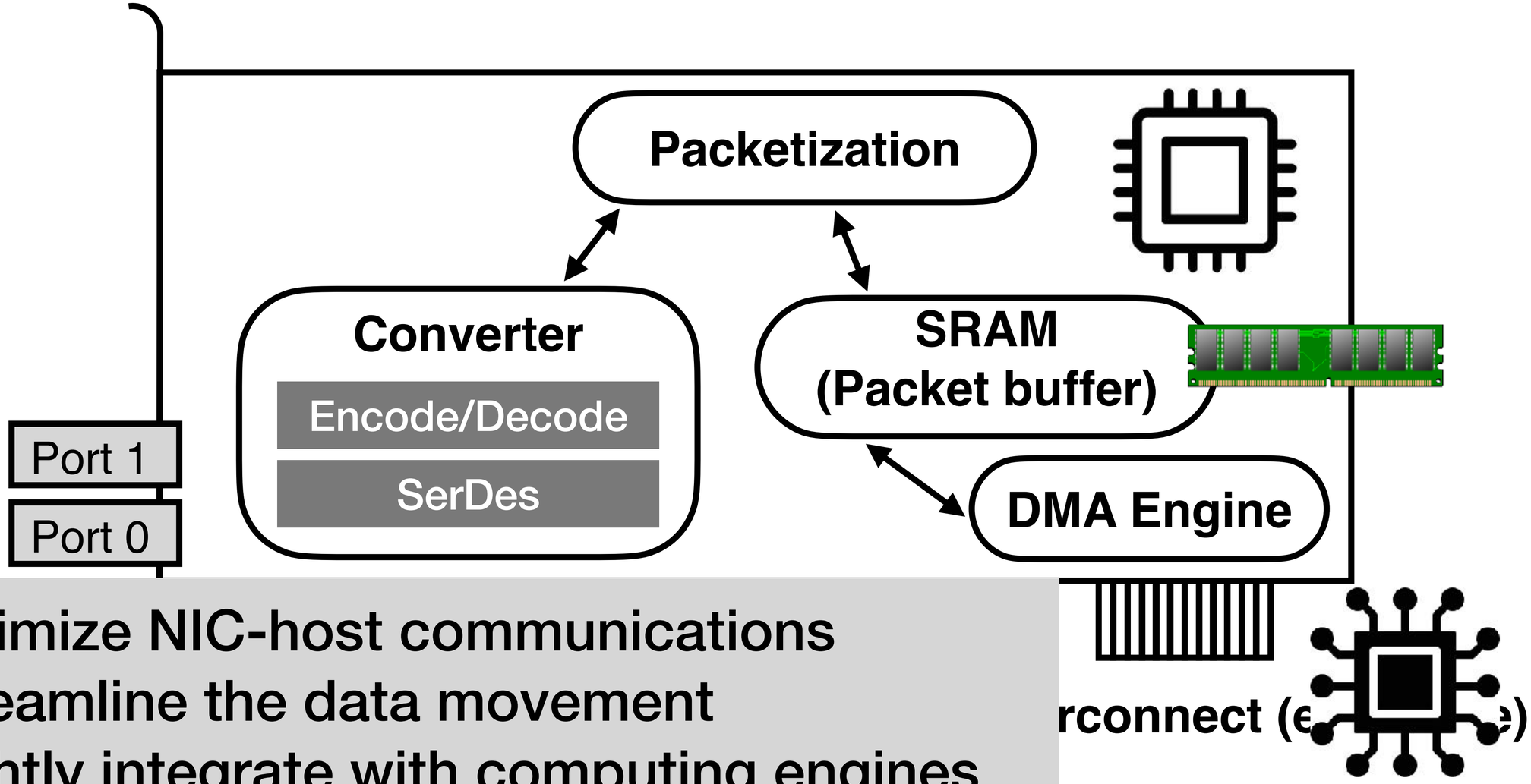
How to make a NIC “Smart”?



#2: Add memory

- Cache-coherent domain
- Exposed read/write port

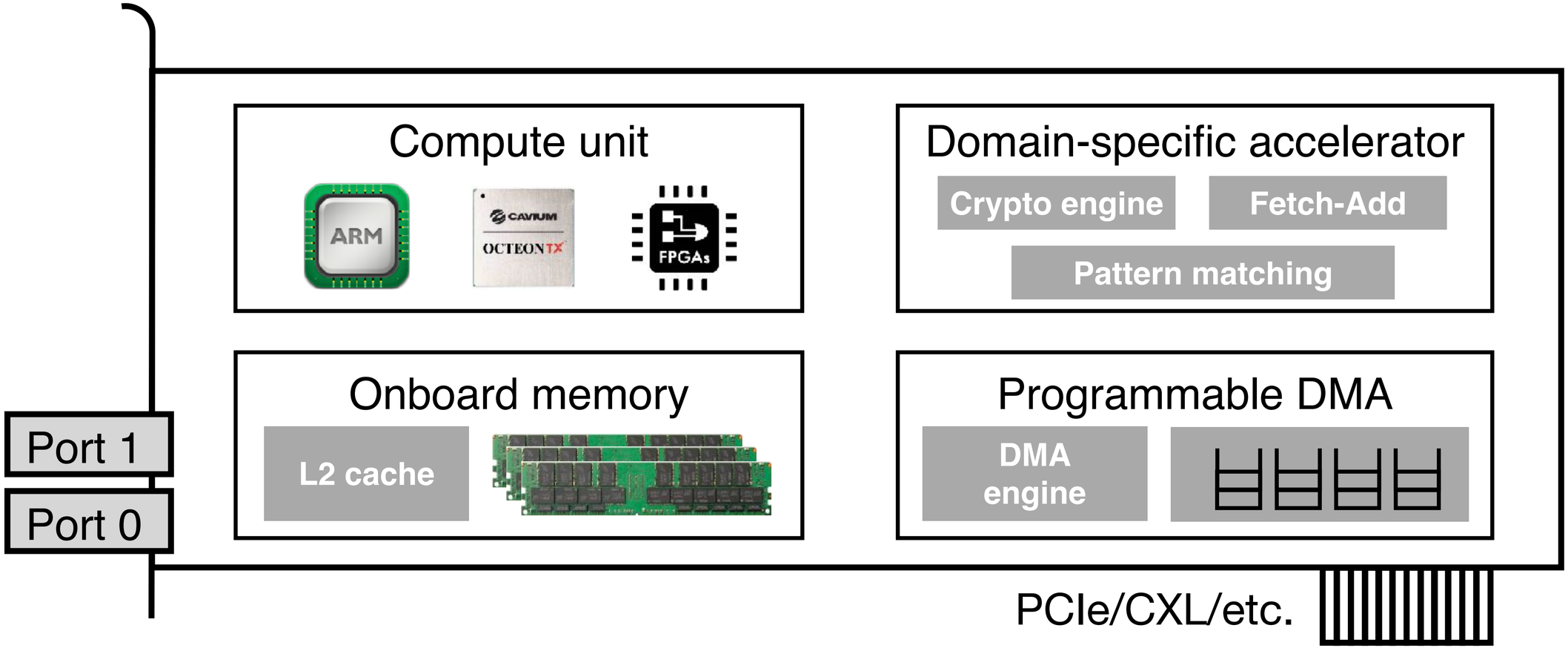
How to make a NIC “Smart”?



#3: Optimize NIC-host communications

- Streamline the data movement
- Tightly integrate with computing engines

SmartNIC


Compute unit
ARM
CAVIUM
OCTEONTX
FPGAs
Domain-specific accelerator
Crypto engine
Fetch-Add
Pattern matching
Onboard memory
L2 cache
Programmable DMA
DMA engine
Port 1
Port 0
PCIe/CXL/etc.

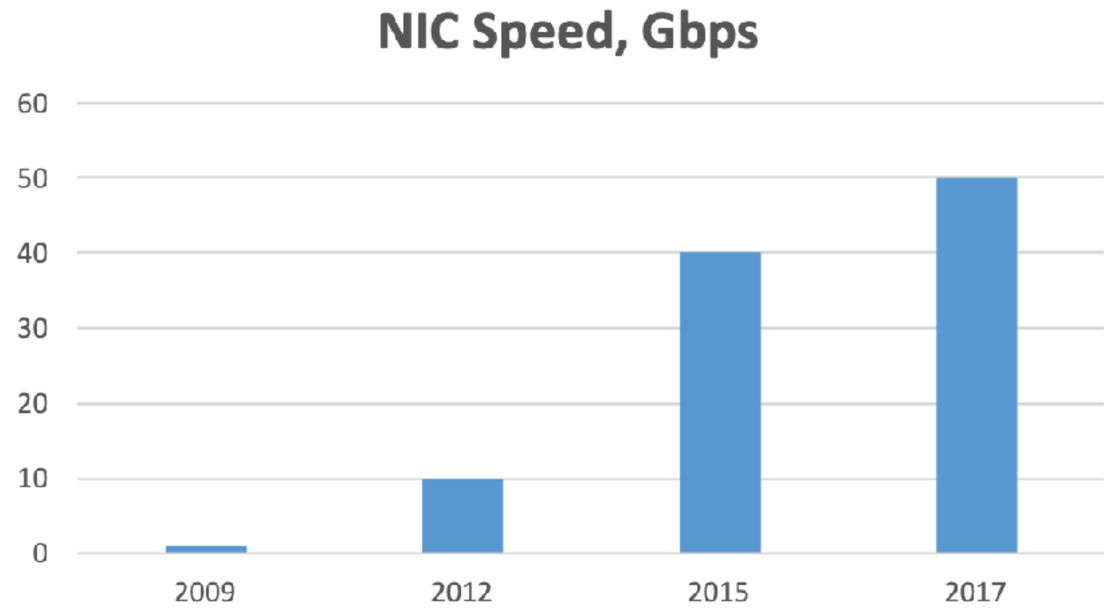
Why SmartNICs?

AccelNet Answer

- Satisfy bandwidth increase with modest CPU cycle consumption

AccelNet Answer

- Satisfy bandwidth increase with modest CPU cycle consumption



We got a 50x improvement in network throughput, but not a a 50x improvement in CPU power!

Why does the SmartNIC run in AccelNet?

Azure VFP

VFP: A Virtual Switch Platform for Host SDN in the Public Cloud

Daniel Firestone, *Microsoft*

Abstract

Many modern scalable cloud networking architectures rely on host networking for implementing VM network policy - e.g. tunneling for virtual networks, NAT for load balancing, stateful ACLs, QoS, and more. We present the Virtual Filtering Platform (VFP) - a programmable virtual switch that powers Microsoft Azure, a large public cloud, and provides this policy. We define several major goals for a programmable virtual switch based on our operational experiences, including support for multiple independent network controllers, policy based on connections rather than only on packets, efficient caching and classification algorithms for performance, and efficient offload of flow policy to programmable NICs, and demonstrate how VFP achieves these goals. VFP has been deployed on >1M hosts running IaaS and PaaS workloads for over 4 years. We present the design of VFP and its API, its flow language and compiler used for flow processing, performance results, and experiences deploying and using VFP in Azure over several years.

1. Introduction

The rise of public cloud workloads, such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform [13-15], has created a new scale of datacenter computing, with vendors regularly reporting server counts in the millions. These vendors not only have to provide scale and the high density/performance of Virtual Machines (VMs) to customers, but must provide rich network semantics, such as private virtual networks with customer supplied address spaces, scalable L4 load balancers, security groups and ACLs, virtual routing tables, bandwidth metering, QoS, and more.

This policy is sufficiently complex that it often cannot economically be implemented at scale in traditional core routers and hardware. Instead a common approach has been to implement this policy in software on the VM hosts, in the virtual switch (vswitch) connecting VMs to the network, which scales well with the number of servers, and allows the physical network to be simple, scalable and very fast. As this model separates a centralized control plane from a data plane on the host, it is widely considered an example of Software Defined Networking (SDN) – in particular, host-based SDN.

As a large public cloud provider, Azure has built its cloud network on host based SDN technologies, using them to implement almost all virtual networking features we offer. Much of the focus around SDN in recent years has been on building scalable and flexible network controllers and services, which is critical. However, the design of the programmable vswitch is equally important. It has the dual and often conflicting requirements of a highly programmable dataplane, with high performance and low overhead, as cloud workloads are cost and performance sensitive.

In this paper, we present the Virtual Filtering Platform, or VFP – our cloud scale virtual switch that runs on all of our hosts. VFP is so named because it acts as a filtering engine for each virtual NIC of a VM, allowing controllers to program their SDN policy. Our goal is to present both our design and our experiences running VFP in production at scale, and lessons we learned.

1.1 Related Work

Throughout this paper, we use two motivating examples from the literature and demonstrate how VFP supports their policies and actions. The first is VL2 [2], which can be used to create virtual networks (VNETs) on shared hardware using stateless tunneling between hosts. The second is Ananta [4], a scalable Layer-4 load balancer, which scales by running the load balancing NAT in the vswitch on end hosts, leaving the in-network load balancers stateless and scalable.

In addition, we make references and comparisons to OpenFlow [5], a programmable forwarding plane protocol, and OpenVswitch [1] (OVS), a popular open source vswitch implementing OpenFlow. These are two seminal projects in the SDN space. We point out core design differences from the perspective of a public cloud on how our constraints can differ from those of open source projects. It is our goal to share these learnings with the broader community.

2. Design Goals and Rationale

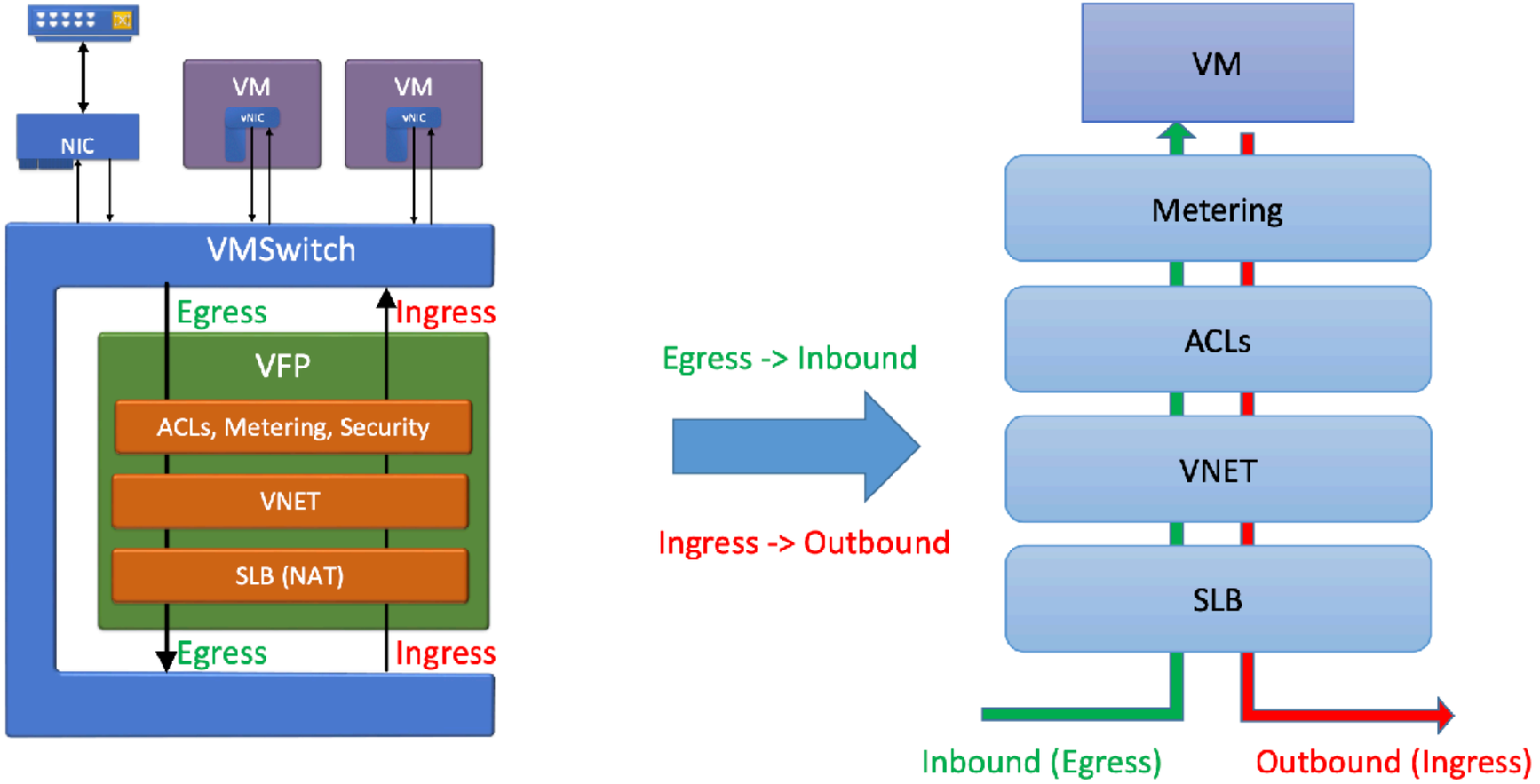
VFP's design has evolved over time based on our experiences running a large public cloud platform. VFP was not our original vswitch, nor were its original functions novel ideas in host networking – VL2 and Ananta already pioneered such use of vswitches.

Originally, we built networking filter drivers on top of Windows's Hyper-V hypervisor for each host function, which we chained together in a vswitch – a stateful firewall driver for ACLs, a tunneling driver for VL2 VNETs, a NAT driver for Ananta load balancing, a QoS driver, etc. As host networking became our main tool for virtualization policy, we decided to create VFP in 2011 after concluding that building new fixed filter drivers for host networking functions was not scalable

VFP translates L2 extensibility (ingress/egress to switch) to L3 extensibility (inbound/outbound to VM).

VFP Overview

- VFP translates L2 extensibility to L3 extensibility
  - Ingress/egress to switch —> Ingress/egress to VM

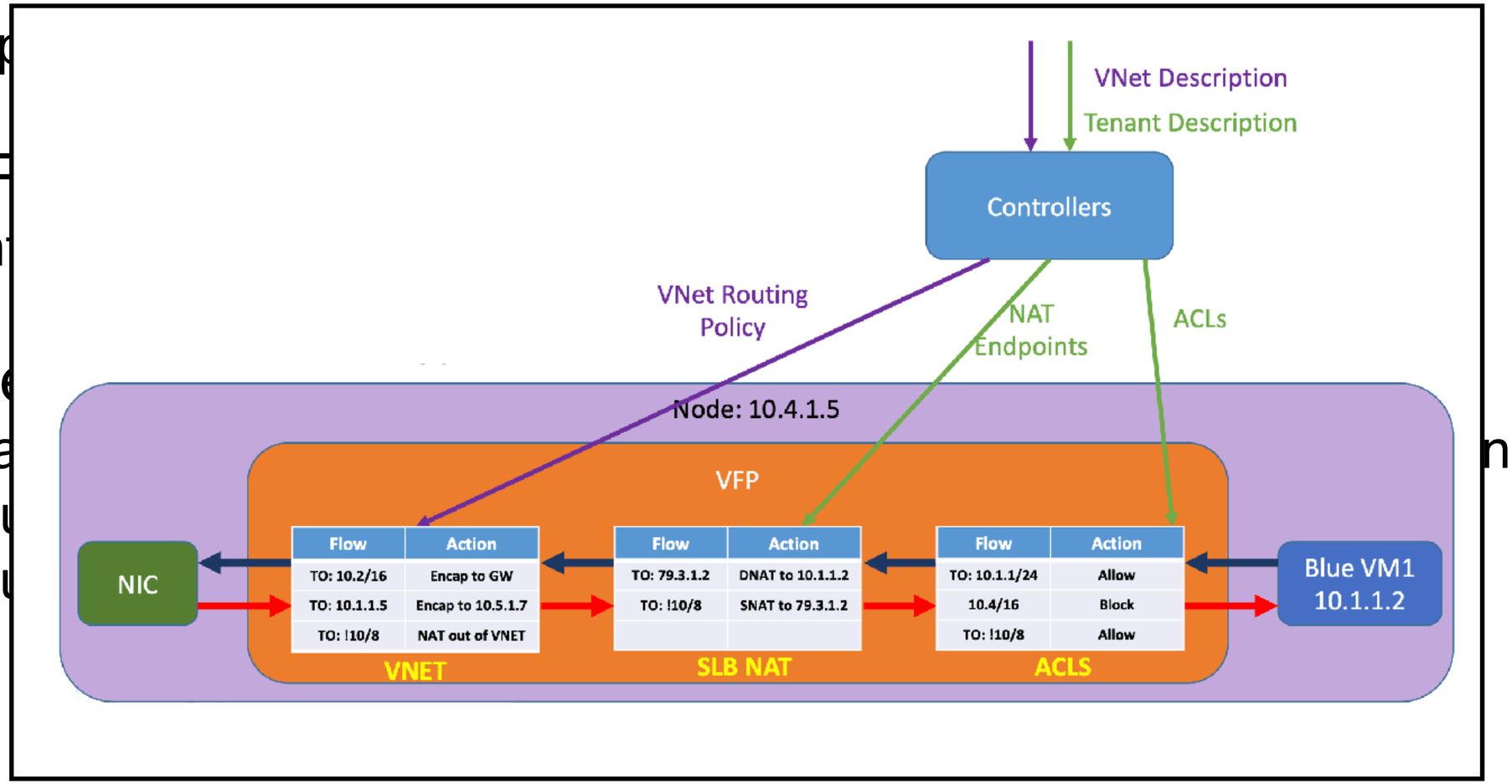


VFP: An OpenFlow-Inspired Design

- VFP controller
  - Specify policies at the flow/packet/VM level
- VFP data-plane
  - Integrate policies and run them on the host side
- Three key primitives
  - Layers: independent flow tables per controller to order the pipeline
  - Rule matches: define which packets match which rule
  - Rule actions: what to do with a packet for a given rule

VFP: An OpenFlow-Inspired Design

- VFP controller
  - Sp
- VFP
  - In
- Thre
  - La
  - Ru
  - Ru

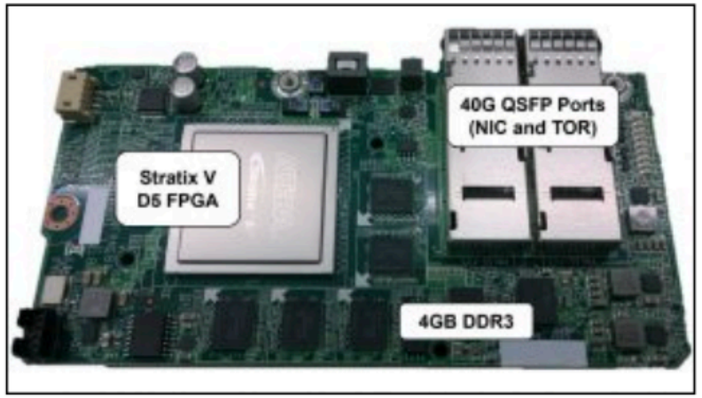


Use Cases

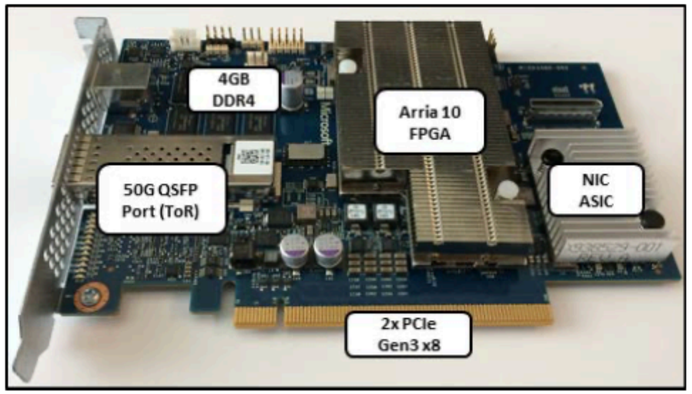
- #1: Cloud Load balancer
- #2: VNet
- #3: 5-tuple ACLs
- #4: Billing
- #5: Rate limiting
- #6: Security guards
- ……

Azure SmartNIC

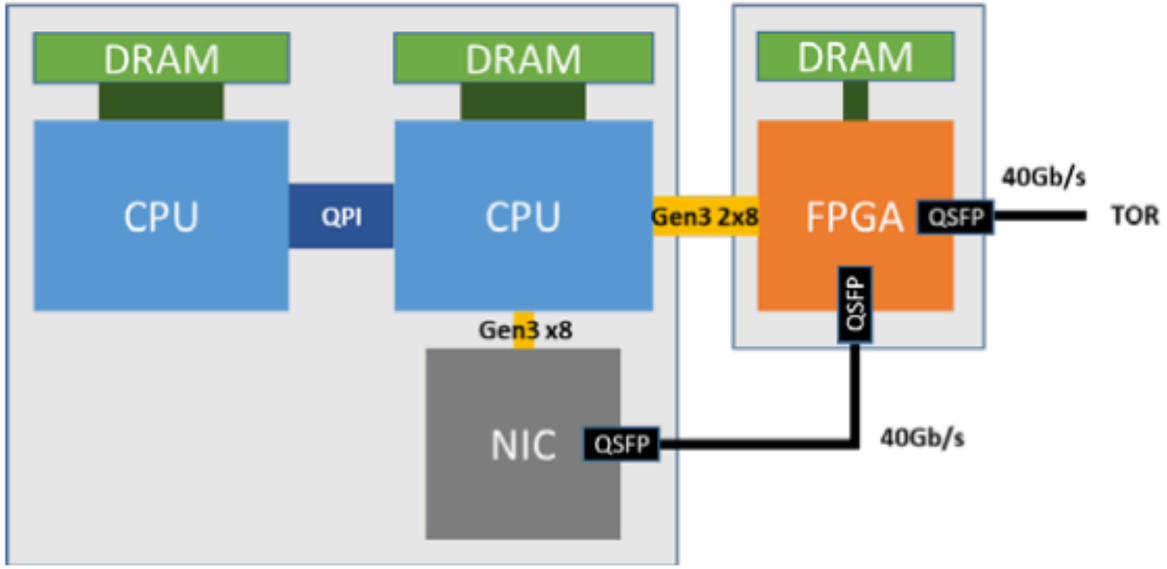
- FPGA-based design
  - Collaborate with Altera



(a) Azure SmartNIC Gen1, 40GbE w/ external NIC



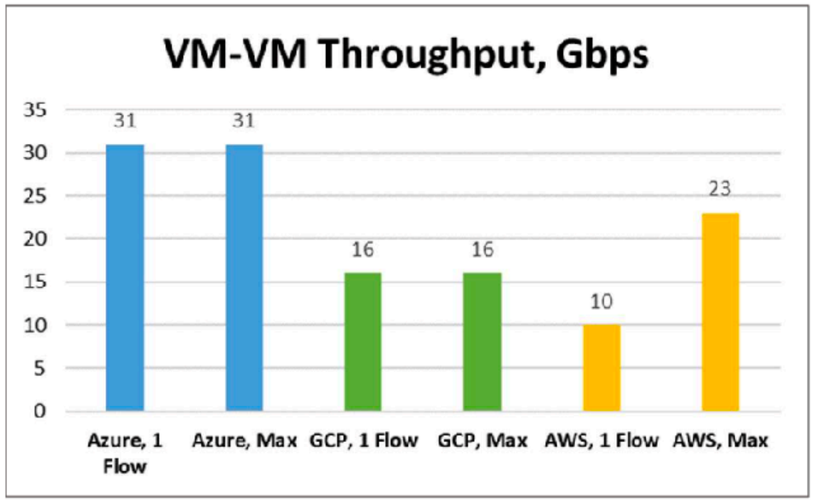
(b) Azure SmartNIC Gen2, 50GbE w/ on-board NIC

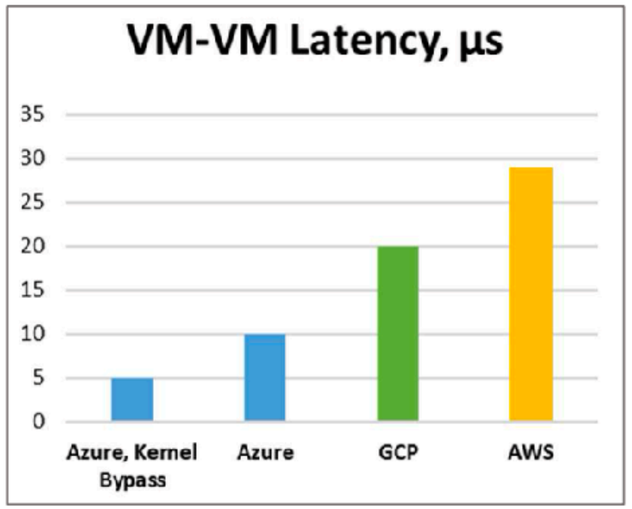


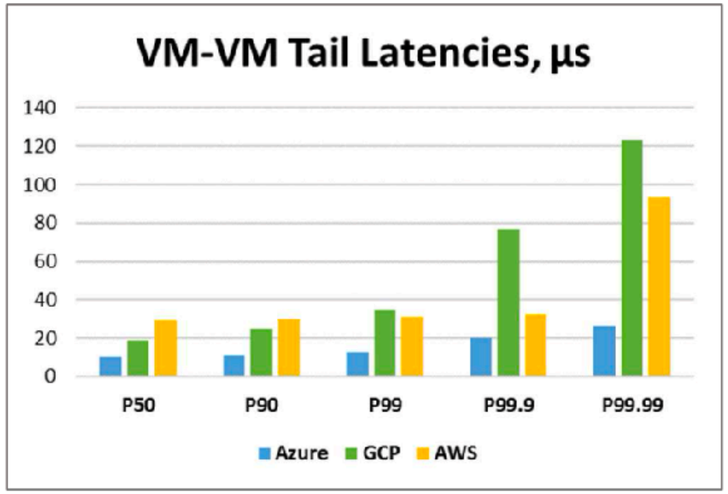
(c) *bump-in-the-wire* architecture

Azure SmartNIC Benefits

- Cost and performance
  - Host core saving + avg/tail latency improvements



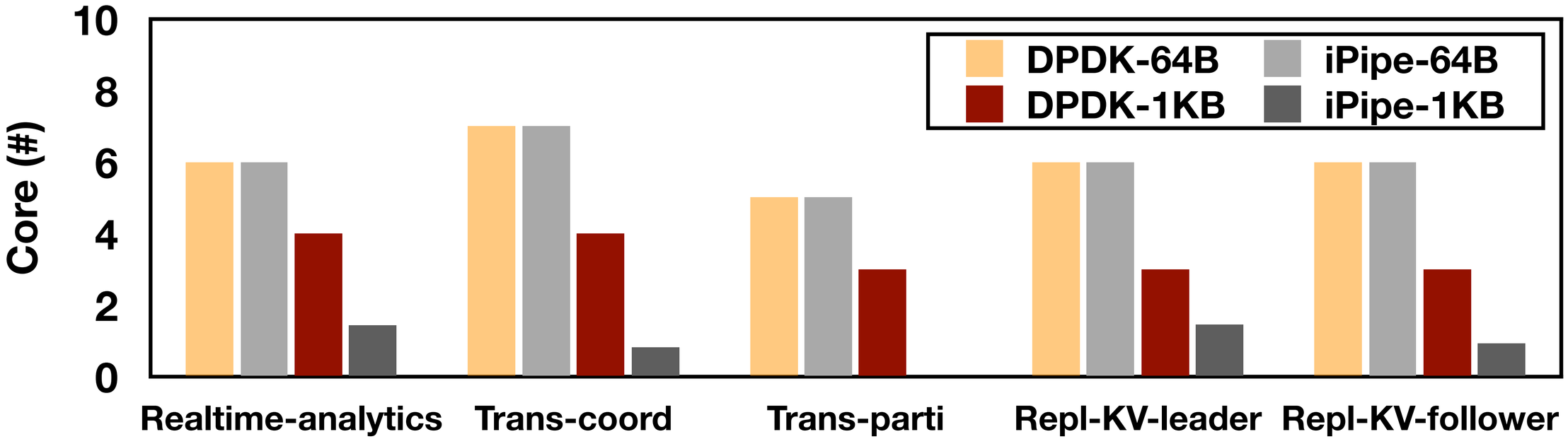




AccelNet demonstrates the benefits from cost and performance perspectives. But FPGA systems are hard to be deployed!

iPipe Answer

- One can achieve similar benefits using commodity SmartNICs
  - Need a new software stack!



SmartNICs Studied in iPipe

- Low power processors with simple micro-architectures
- Varying level of systems support (firmware to Linux)
- Some support RDMA & DPDK interfaces

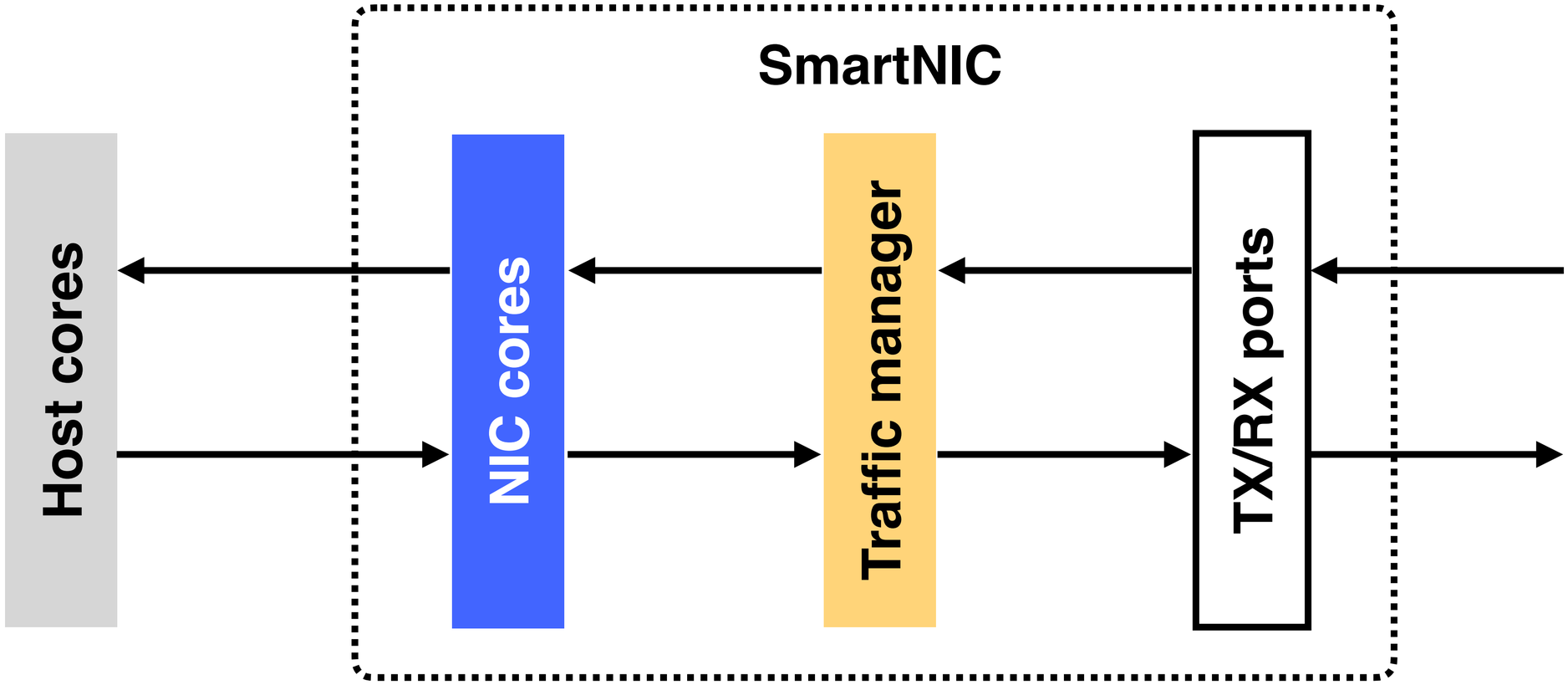
	Vendor	BW	Processor	Deployed SW
LiquidIOII CN2350	Marvell	2X 10GbE	12 cnMIPS core 1.2GHz	Firmware
LiquidIOII CN2360	Marvell	2X 25GbE	16 cnMIPS core, 1.5GHz	Firmware
BlueField 1M332A	Mellanox	2X 25GbE	8 ARM A72 core, 0.8GHz	Full OS
Stingray PS225	Broadcom	2X 25GbE	8 ARM A72 core, 3.0GHz	Full OS

Structural Differences

- Classified into two types based on packet flow
  - On-path SmartNICs
  - Off-path SmartNICs

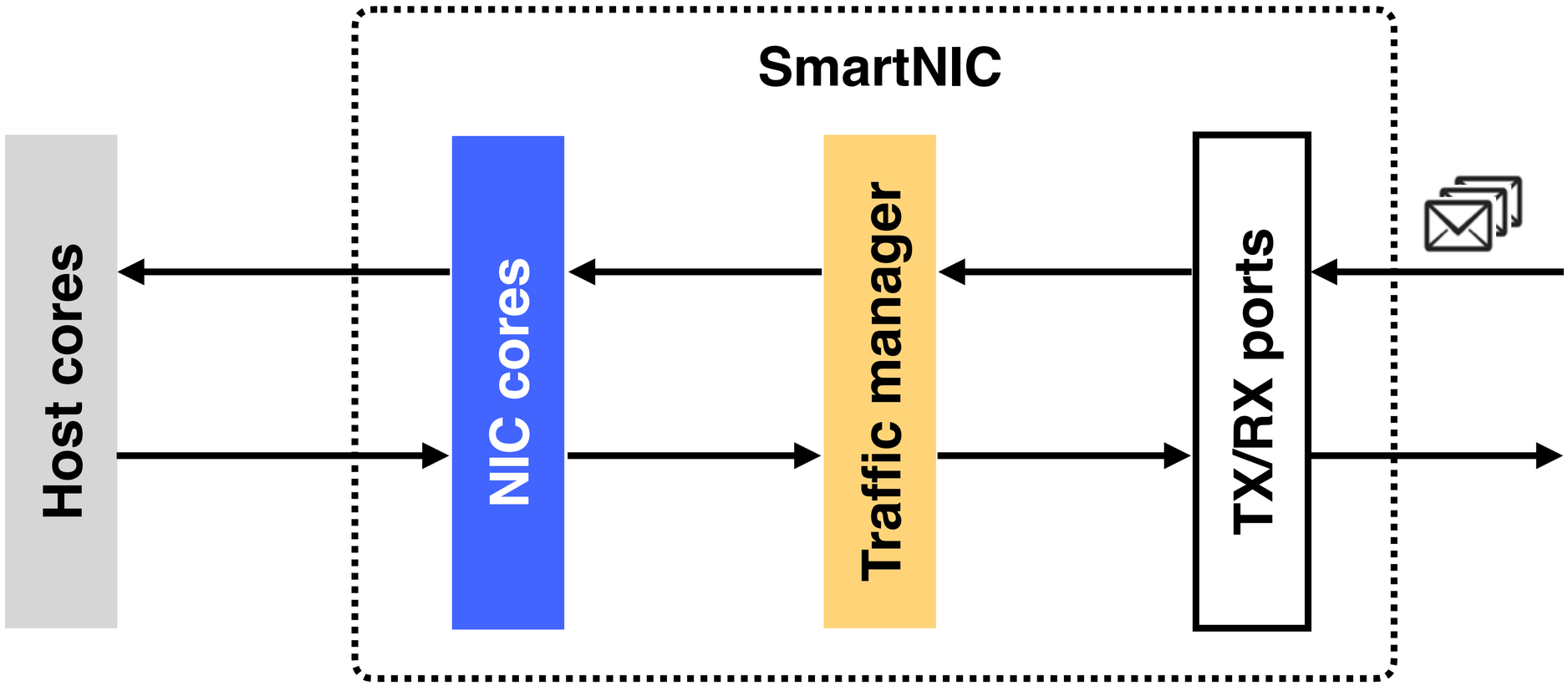
On-path SmartNICs

- NIC cores handle all traffic on both the send & receive paths



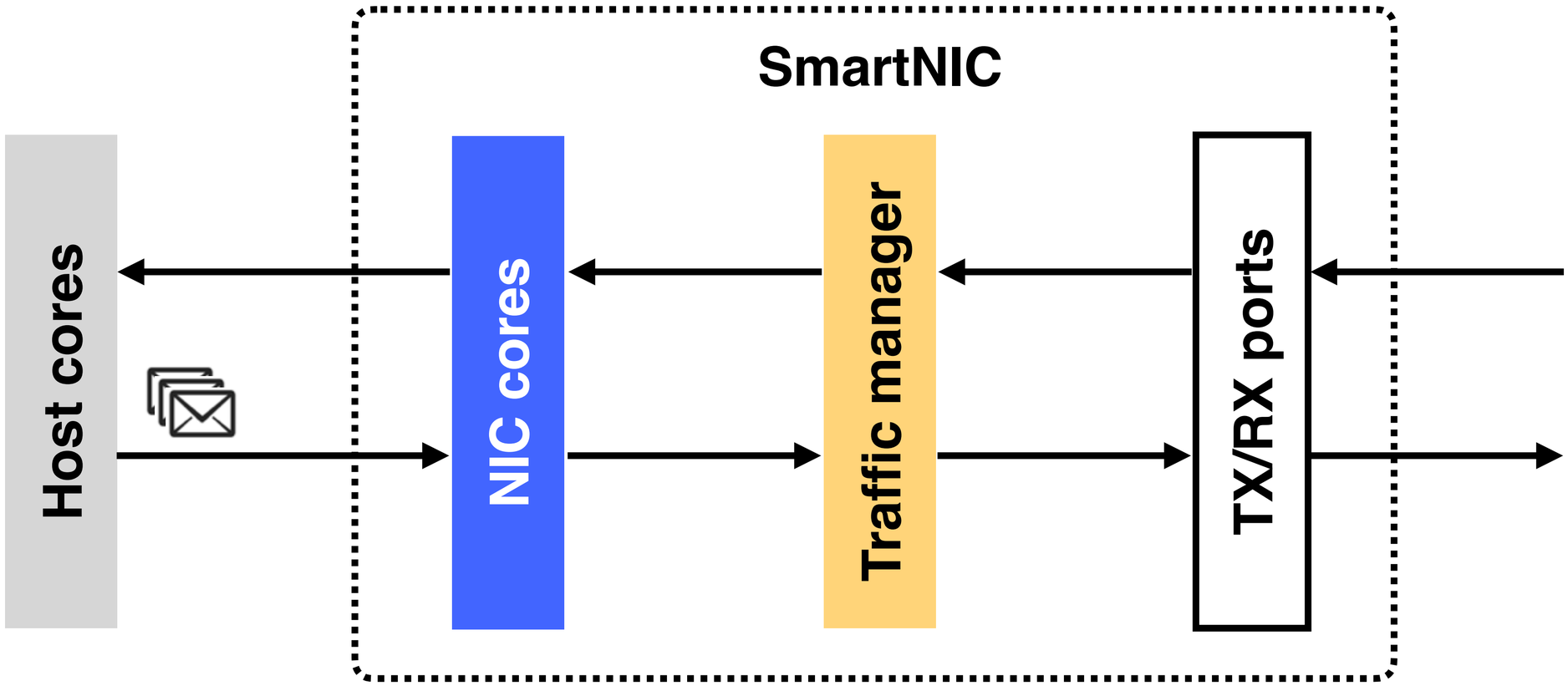
On-path SmartNICs: receive path

- NIC cores handle all traffic on both the send & receive paths



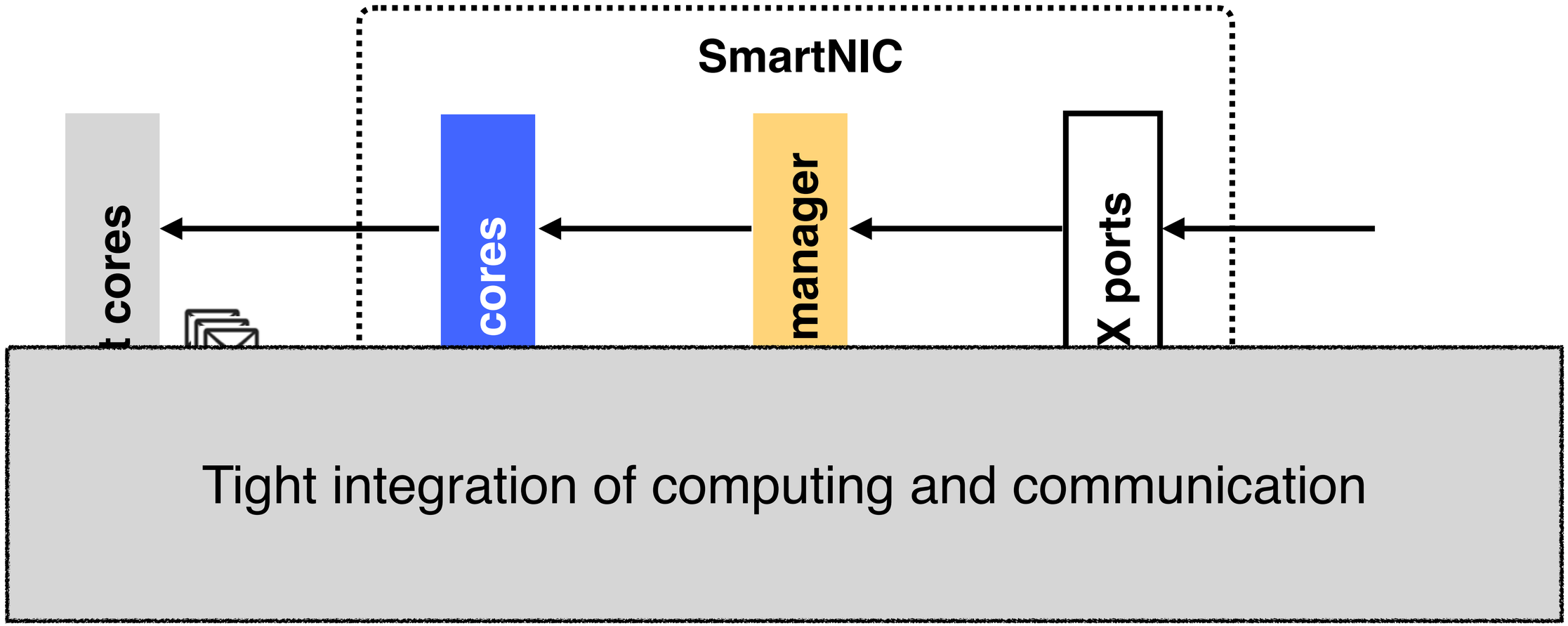
On-path SmartNICs: send path

- NIC cores handle all traffic on both the send & receive paths



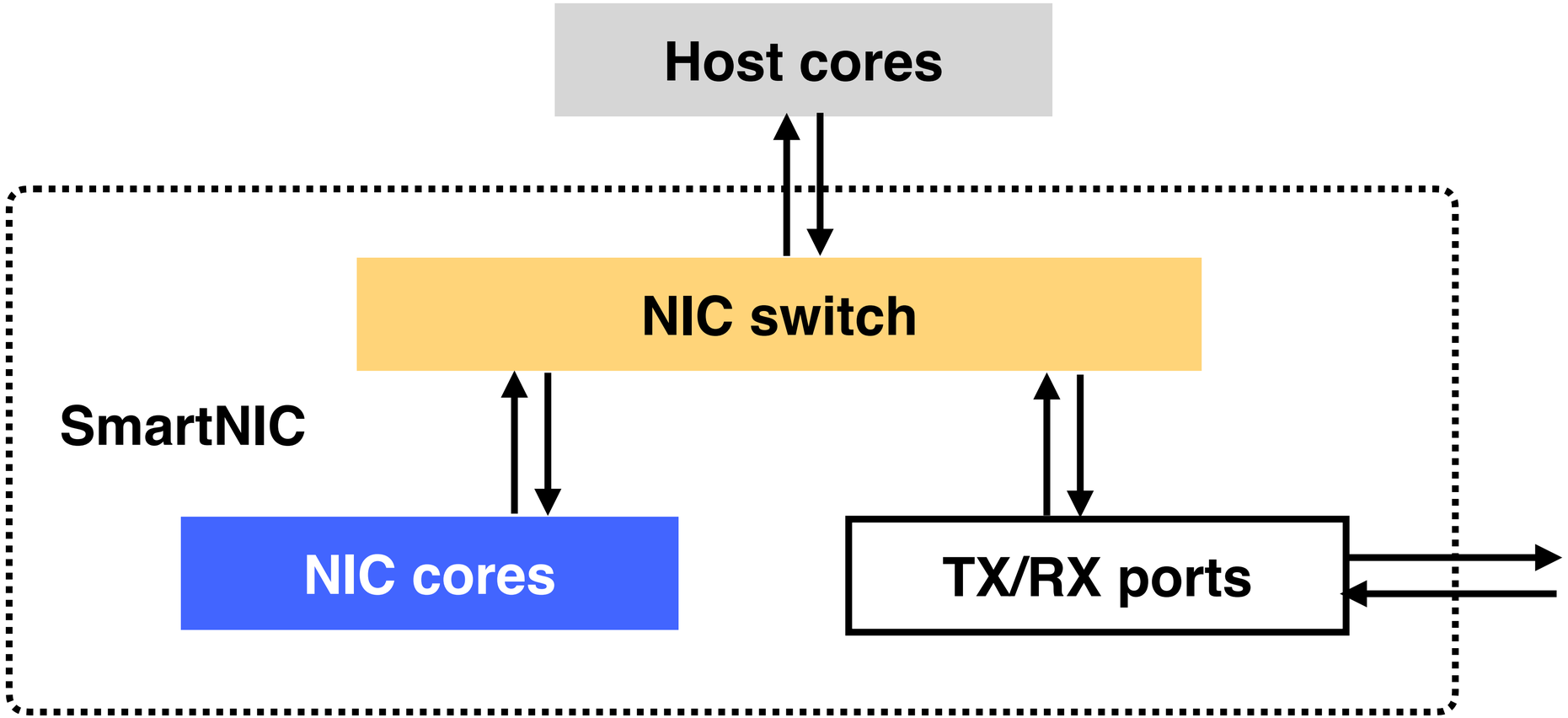
On-path SmartNICs: send path

- NIC cores handle all traffic on both the send & receive paths



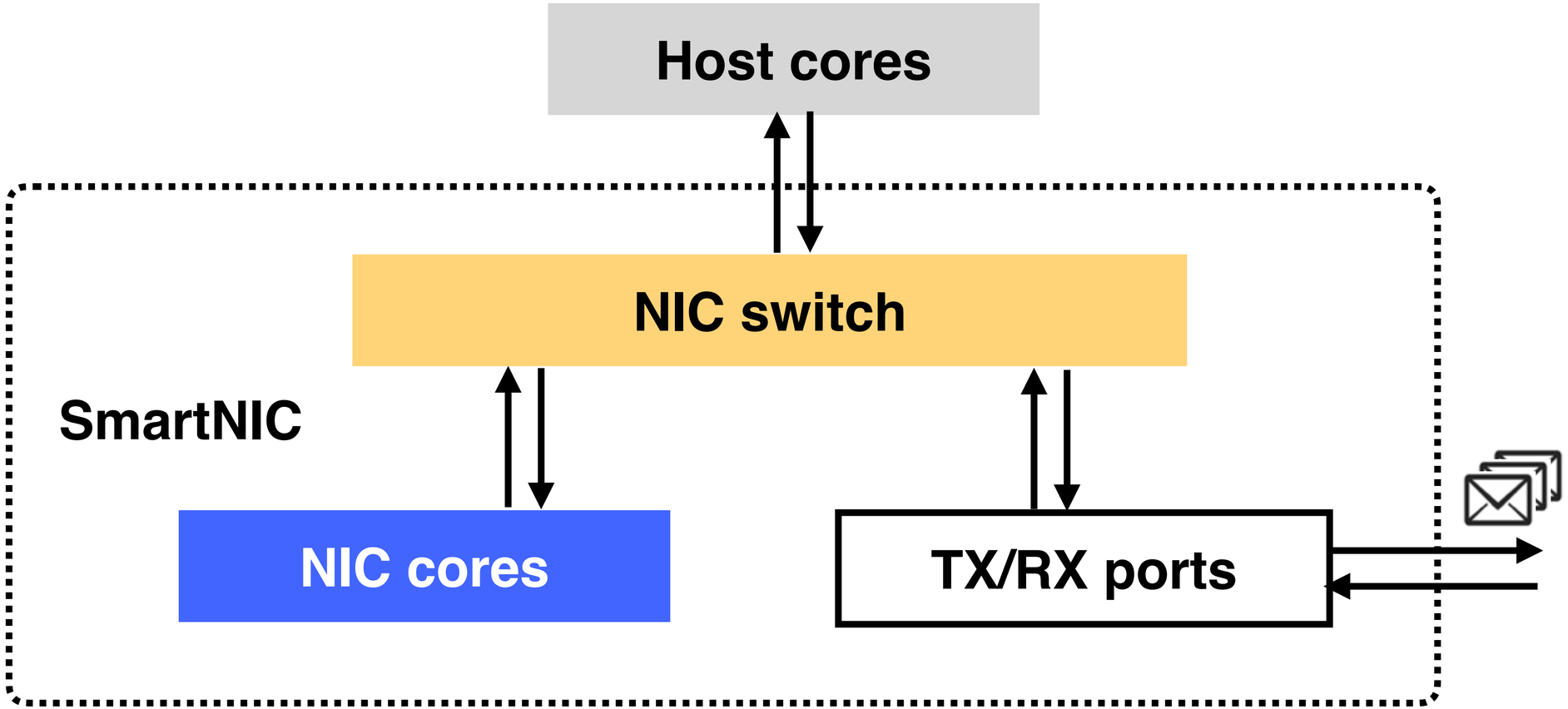
Off-path SmartNICs

- Programmable NIC switch enables targeted delivery



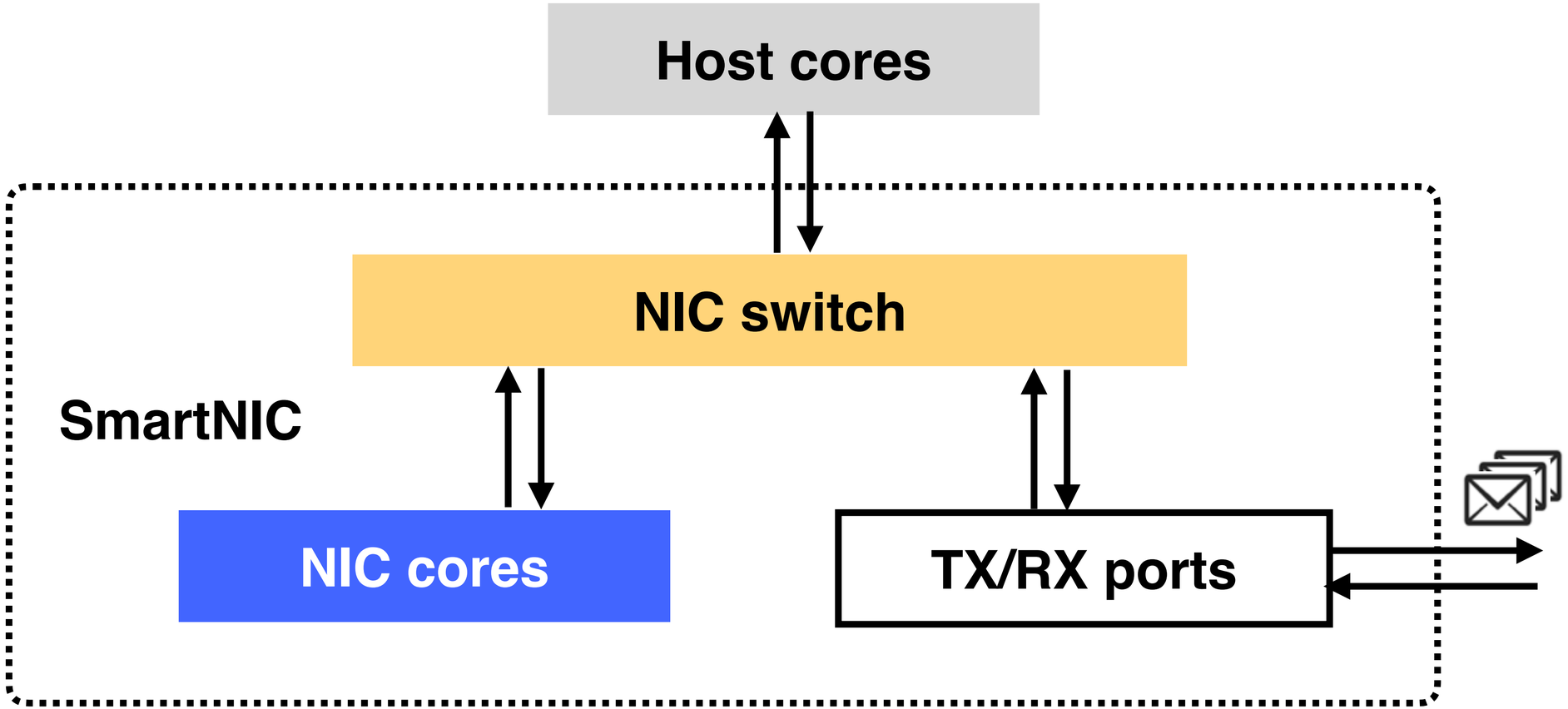
Off-path SmartNICs: receive path

- Programmable NIC switch enables targeted delivery



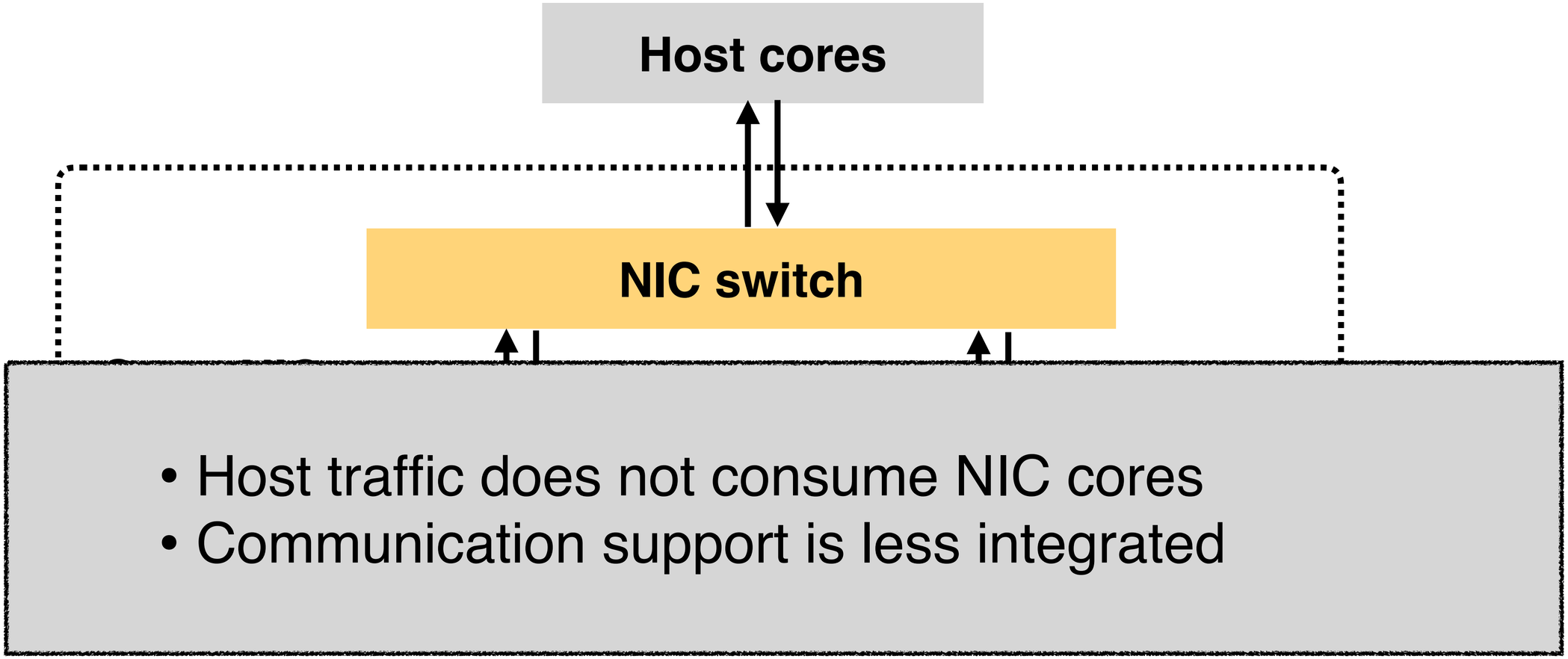
Off-path SmartNICs: receive path

- Programmable NIC switch enables targeted delivery



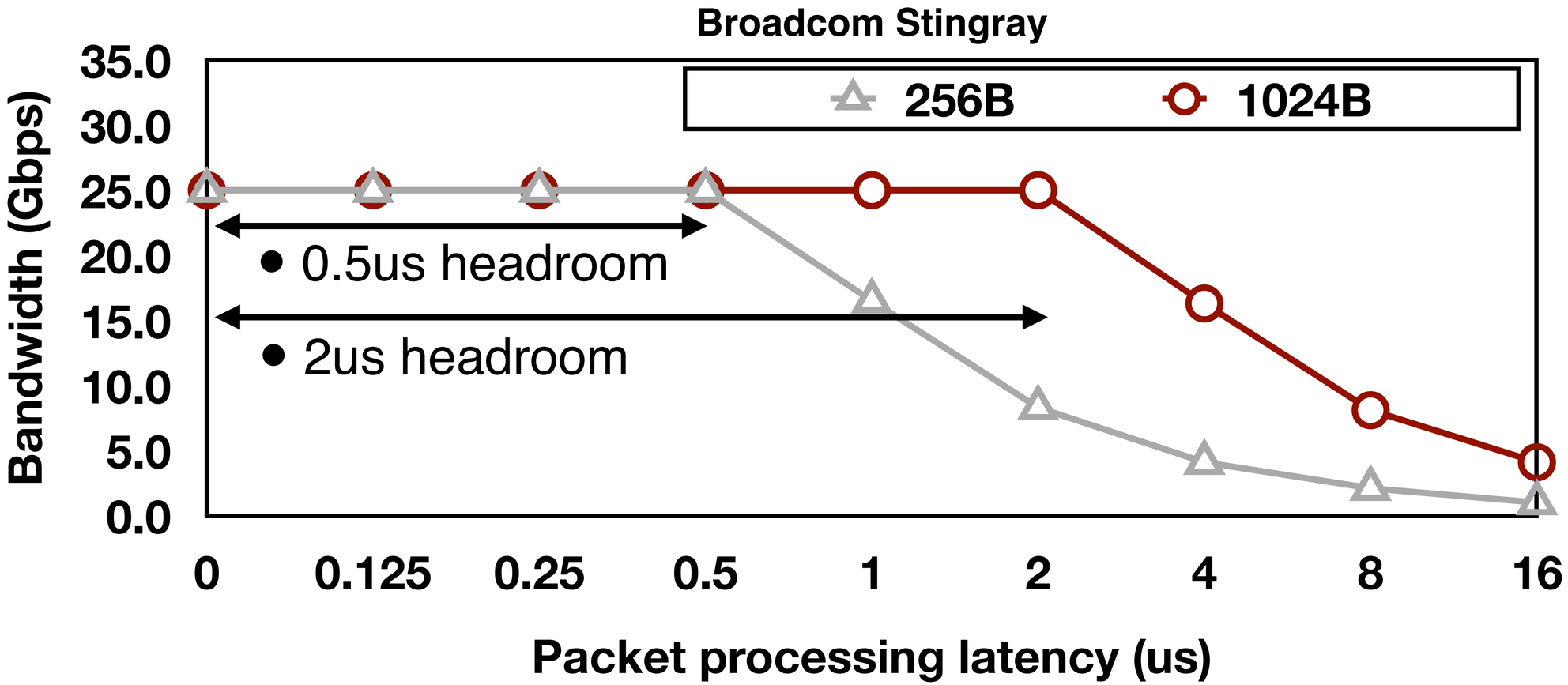
Off-path SmartNICs

- Programmable NIC switch enables targeted delivery



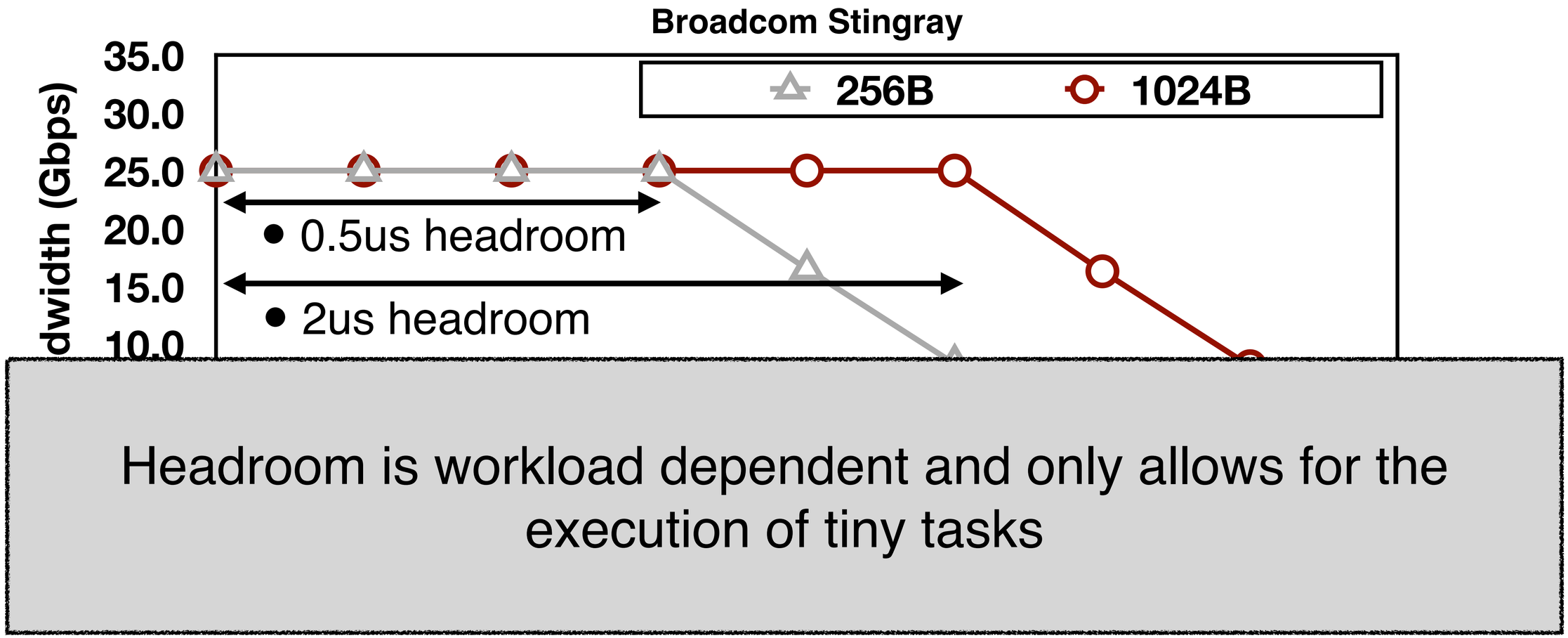
What is the processing headroom on SmartNICs?

- Measure communication throughput upon introducing additional per-packet processing



What is the processing headroom on SmartNICs?

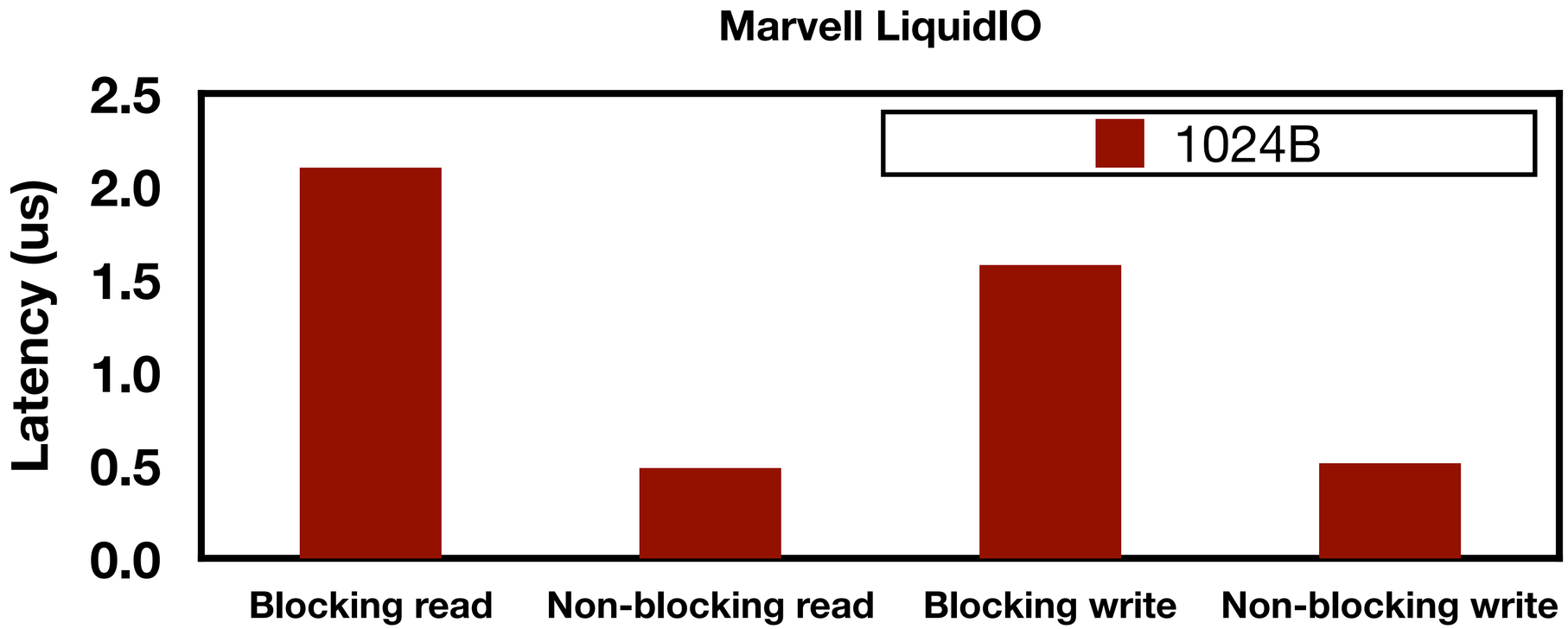
- Measure communication throughput upon introducing additional per-packet processing



Headroom is workload dependent and only allows for the execution of tiny tasks

What is the cost of communicating to the host?

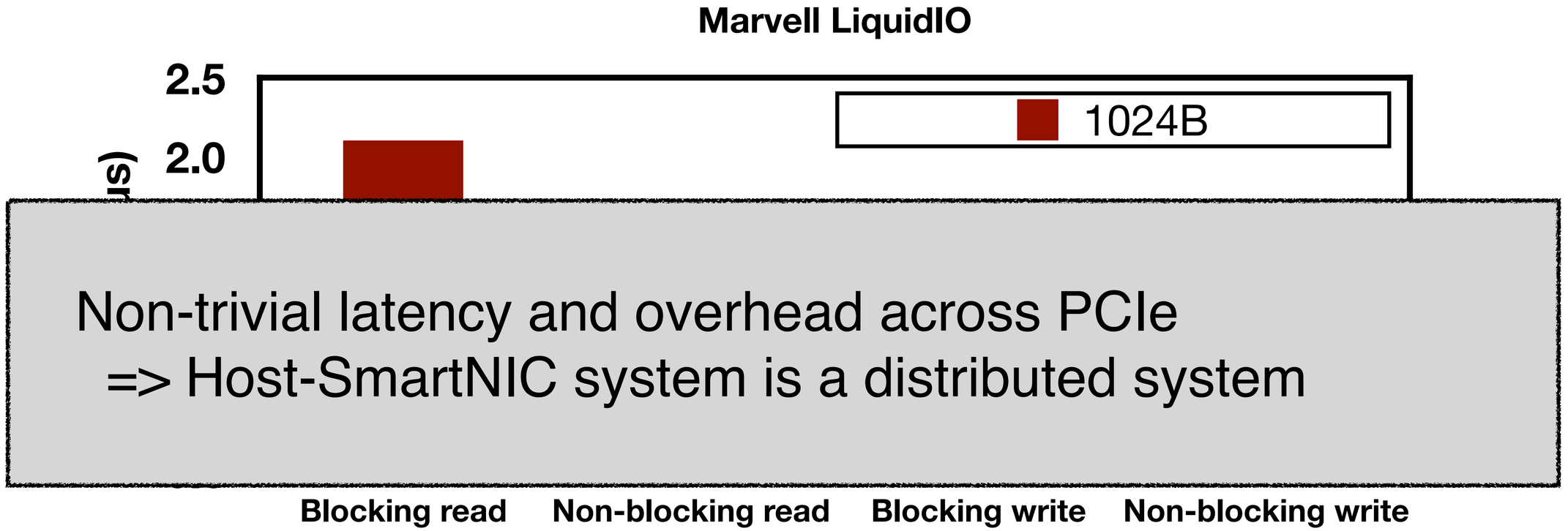
- Traverse PCIe bus either through low-level DMA or higher-level RDMA/ DPDK interfaces



- Read/Write latency for 1024B: 1.5~2us; Overhead: 0.5us

What is the cost of communicating to the host?

- Traverse PCIe bus either through low-level DMA or higher-level RDMA/ DPDK interfaces



- Read/Write latency for 1024B: 1.5~2us; Overhead: 0.5us

iPipe framework

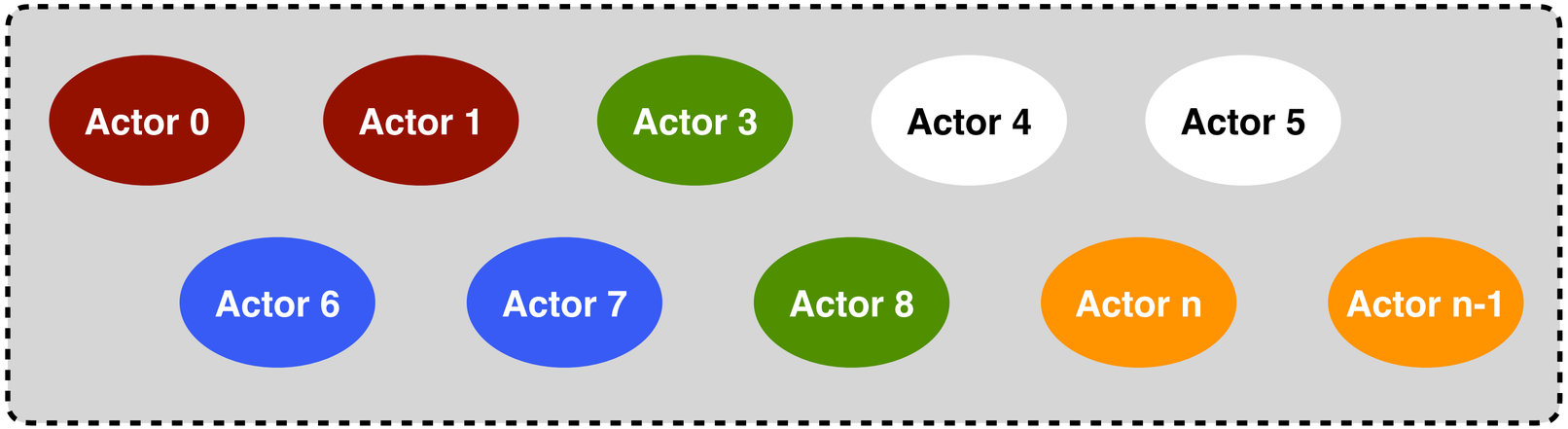
- Goals:
  - Programming framework offloading distributed applications to SmartNIC
  - Addresses the challenges identified by our experiments

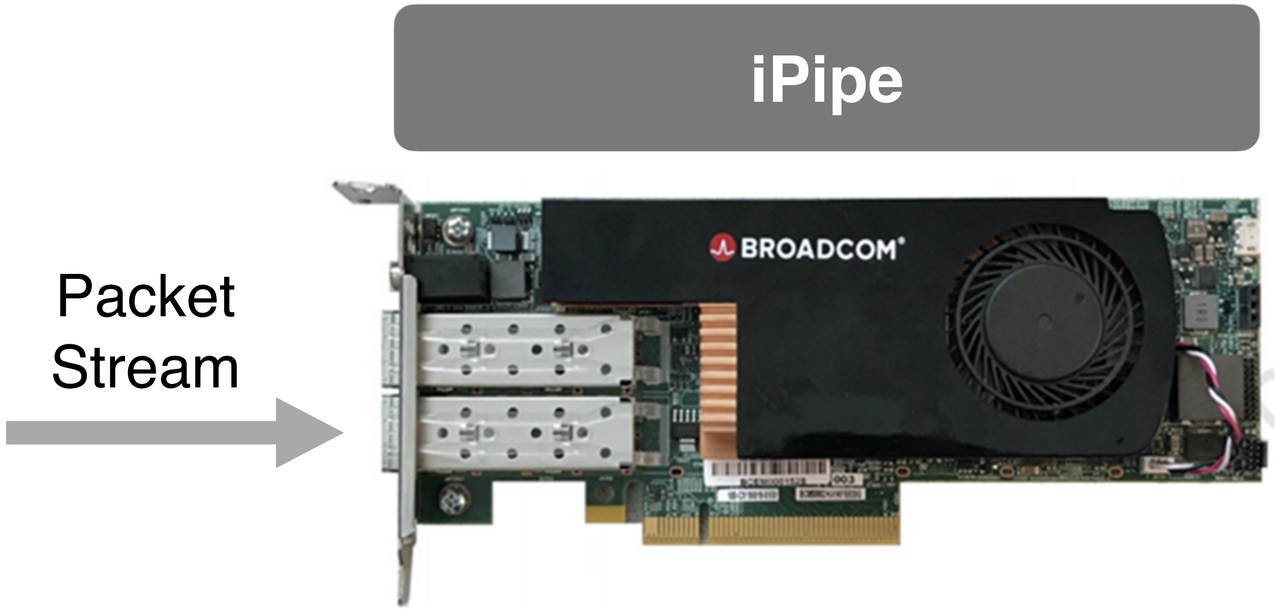
iPipe framework

- Goals:
  - Programming framework offloading distributed applications to SmartNIC
  - Addresses the challenges identified by our experiments

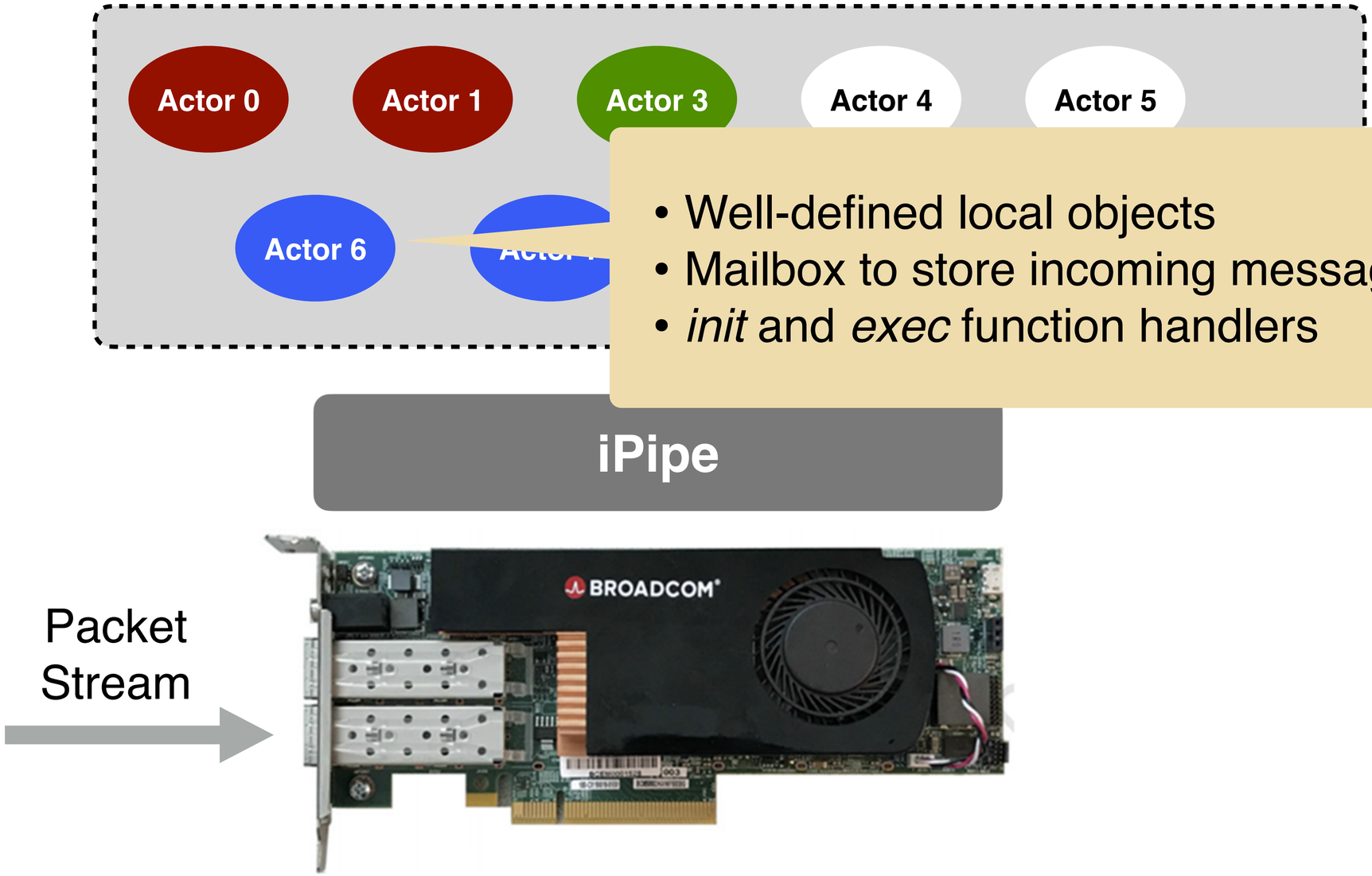
- Host communication overheads => distributed actors
- Variations in traffic workloads => dynamic migration
- Variations in execution costs => scheduler for tiny tasks

Actor programming model

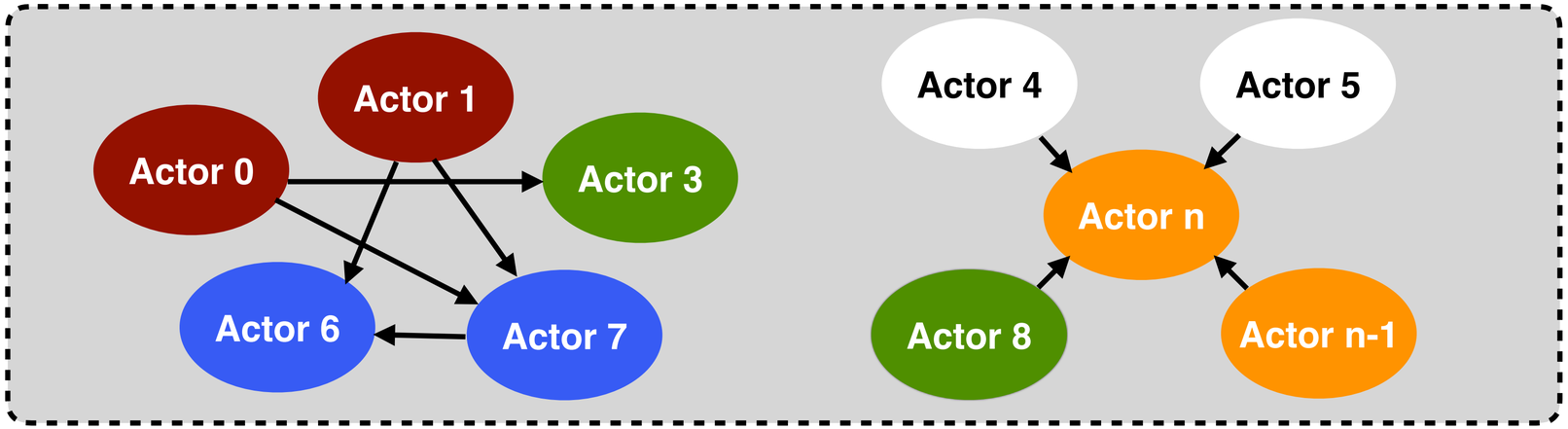




Actor programming model



Actor programming model



iPipe

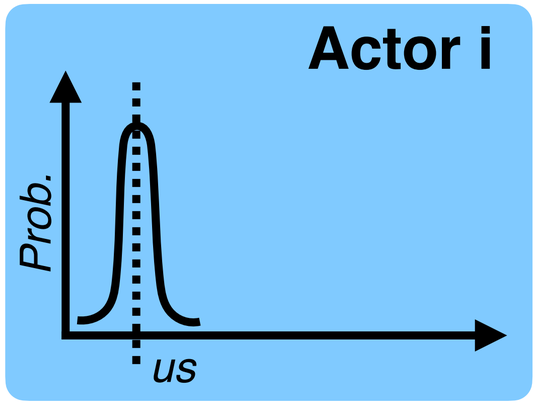
Packet Stream

Actor scheduler

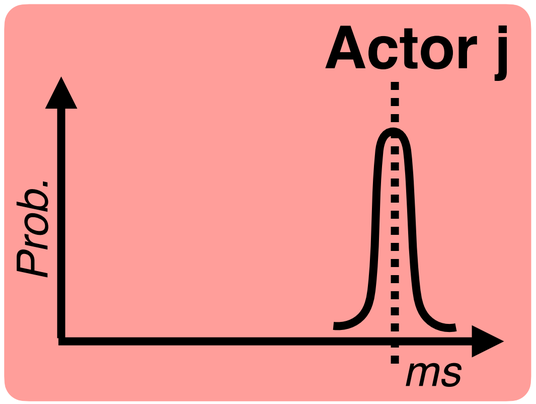
- Goal is to maximize SmartNIC usage, and
  - Prevent overloading and ensure line-rate communications
  - Provide isolation and bound tail latency for actor tasks

Actor scheduler

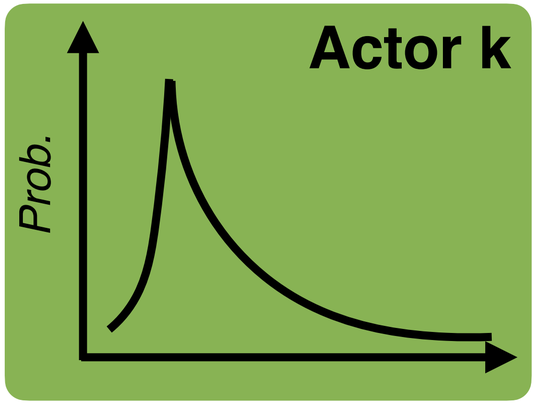
- Goal is to maximize SmartNIC usage, and
  - Prevent overloading and ensure line-rate communications
  - Provide isolation and bound tail latency for actor tasks



- **Low dispersion**
- **Microsecond**



- **Low dispersion**
- **Millisecond**



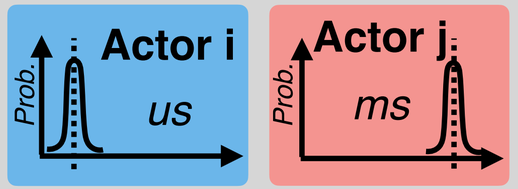
- **High dispersion**
- **Microsecond**

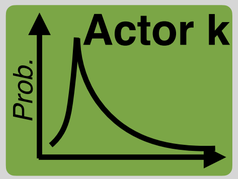
Actor scheduler

- Goal is to maximize SmartNIC usage, and
  - Prevent overloading and ensure line-rate communications
  - Provide isolation and bound tail latency for actor tasks

Theoretical basis:

- Shortest Job First (SJF) optimizes mean response time for arbitrary task distributions
- If the tail response time is to be optimized:
  - First-come first-served (FCFS) is optimal for low variance tasks
  - Processor sharing (PS) is optimal for high variance tasks





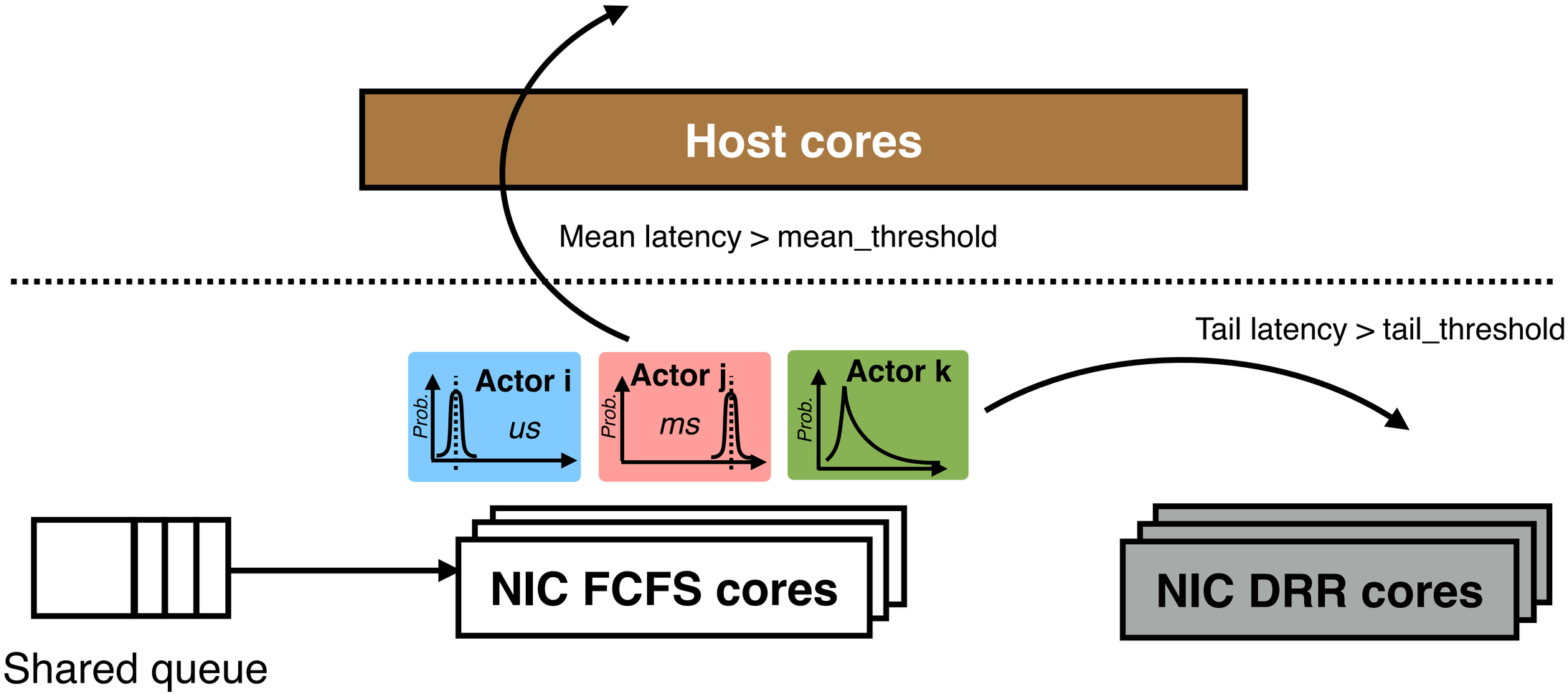
iPipe’s hybrid scheduler

- Combine FCFS and deficit round robin (DRR)
  - Use FCFS to serve tasks with low variance in service times
  - DRR approximates PS in a non-preemptible setting

- Dynamically change actor location & service discipline
  - Monitor bounds on aggregated mean and tail latencies
  - Profile the mean and tail latency of actor innovations

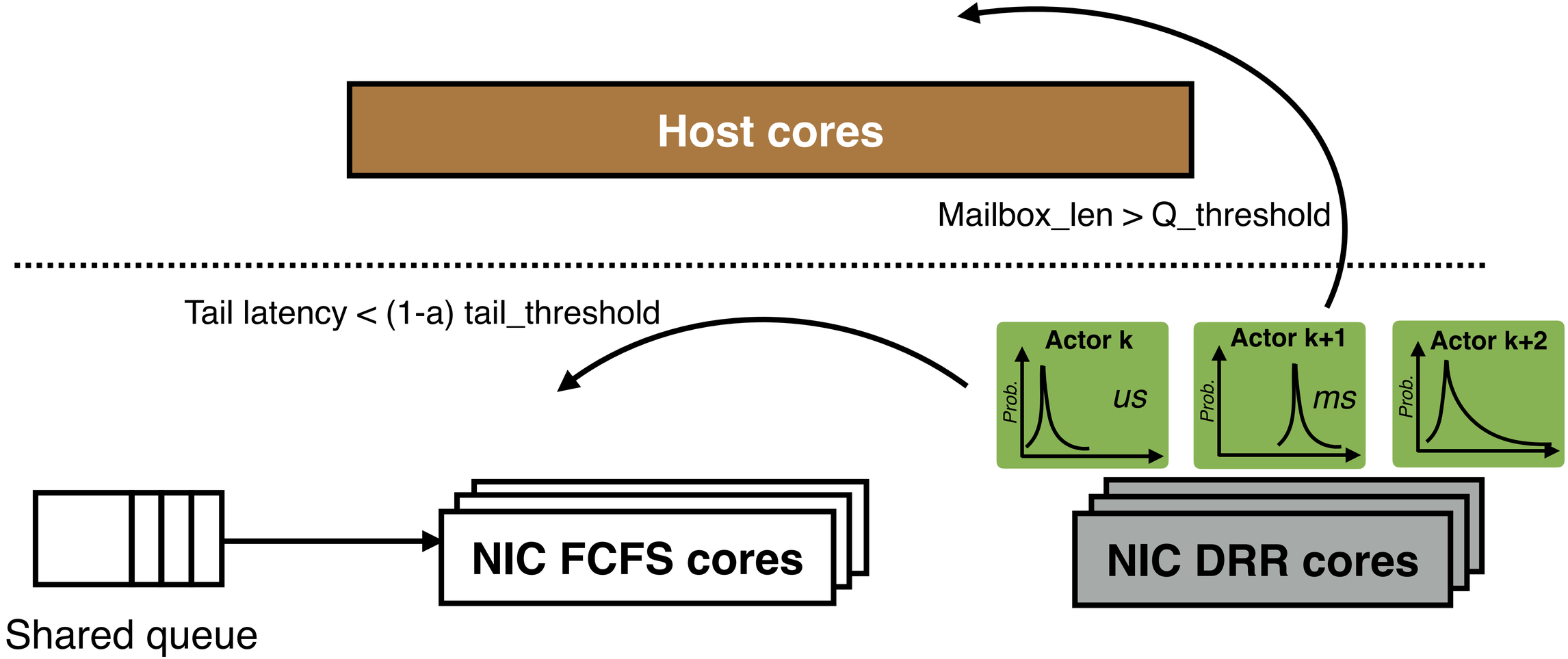
FCFS scheduling

- FCFS cores fetch incoming requests from a shared queue and perform run-to-completion execution



DRR scheduling

- DRR cores traverse the runnable queue and execute actor when its deficit counter is sufficiently high



Applications built using iPipe

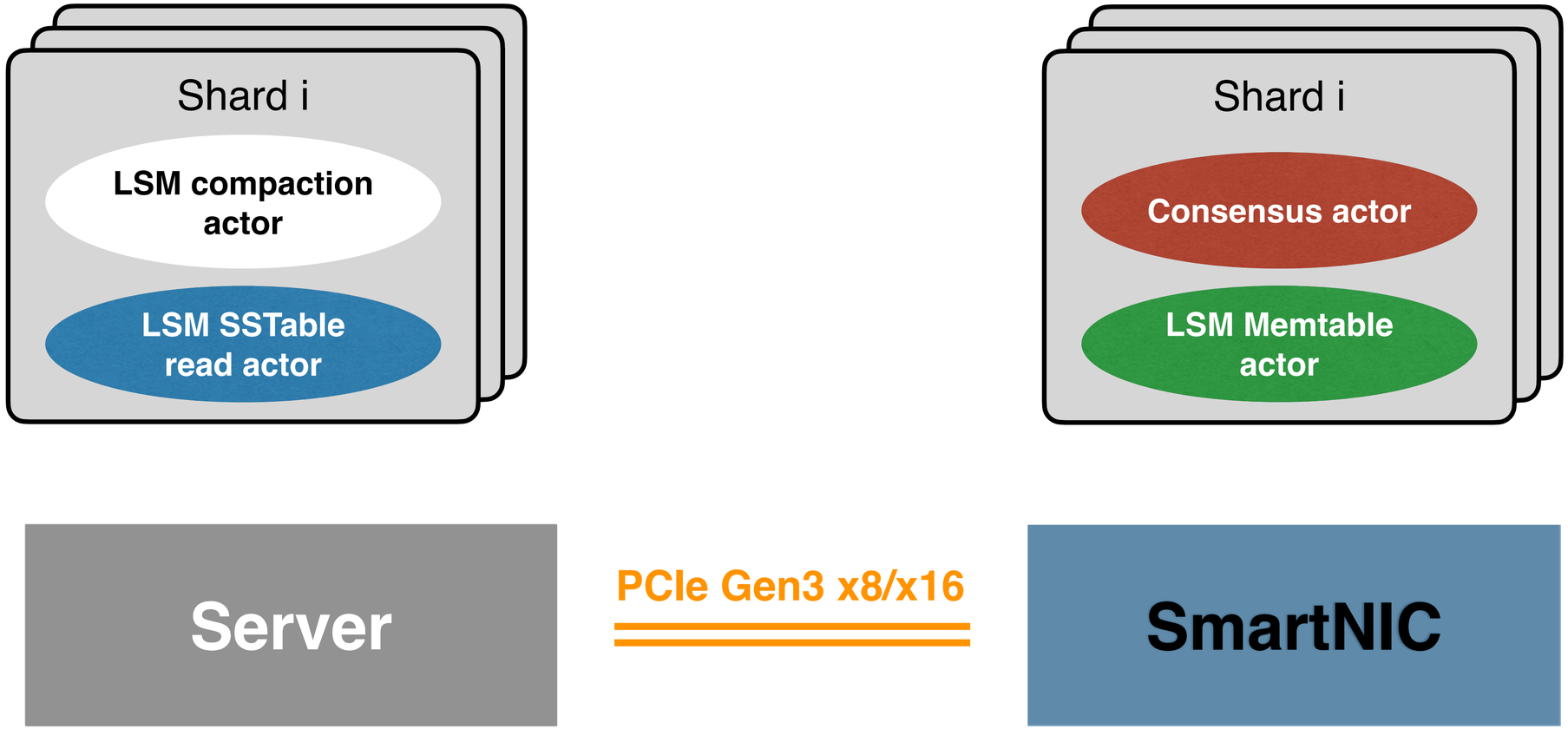
- Replicated and consistent key-value store
- Real time analytics
- Transaction processing system

Replicated key-value store on iPipe

- Consensus layer: Multi-Paxos protocol
- Data store layer: Log-structured merge-tree

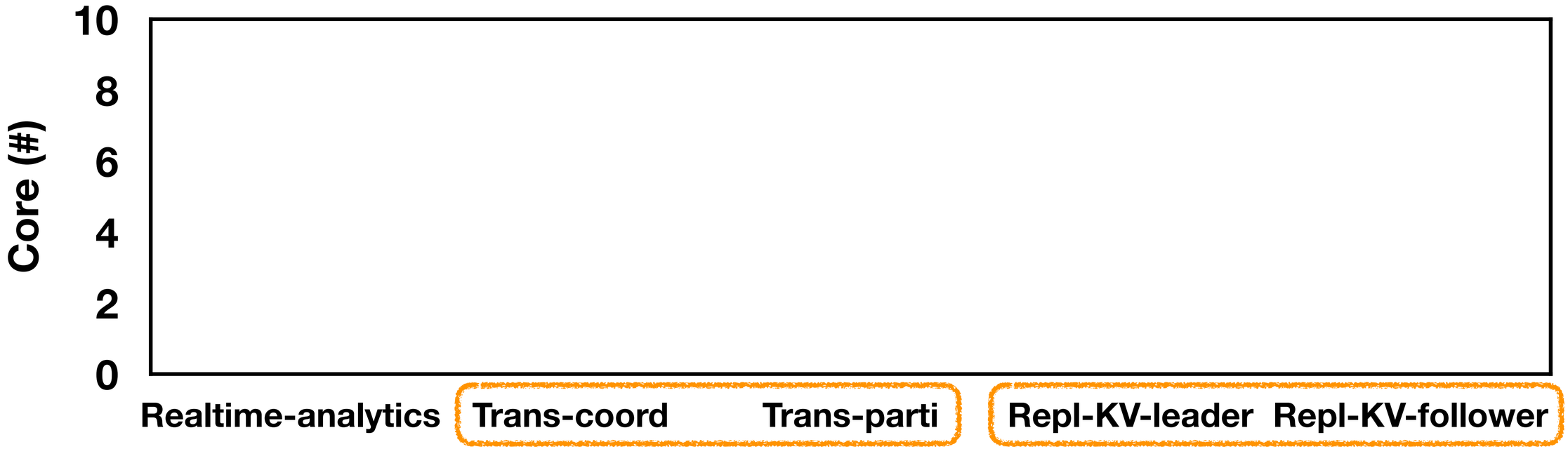
Replicated key-value store on iPipe

- Memtable/commit log is typically resident on SmartNICs
- Compaction and serialization operations on the host

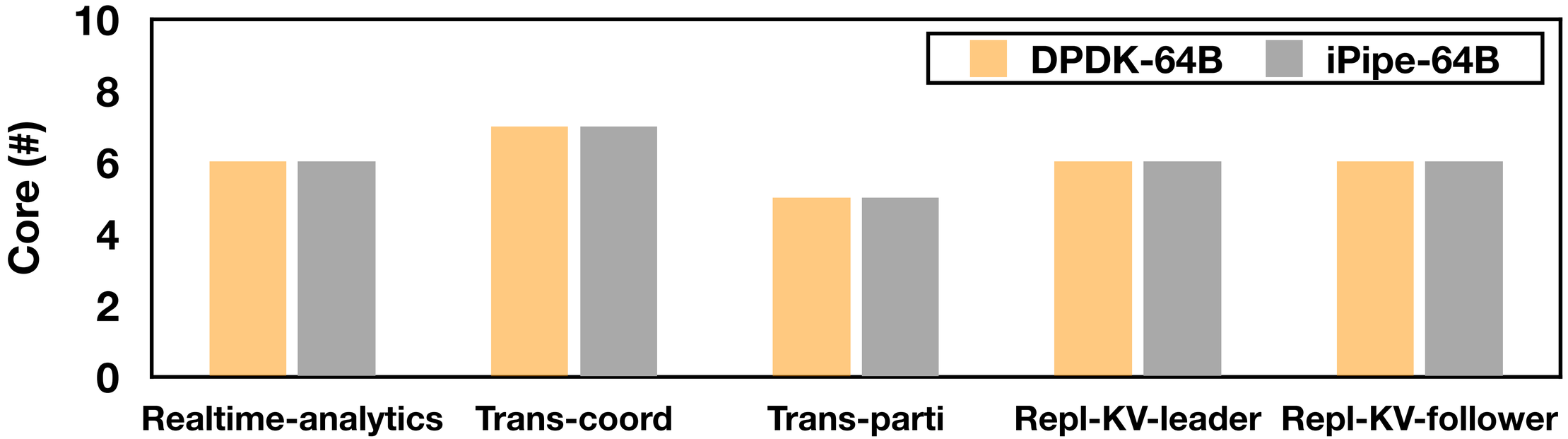


Host core savings for 25Gbps LiquidIOII CN2360

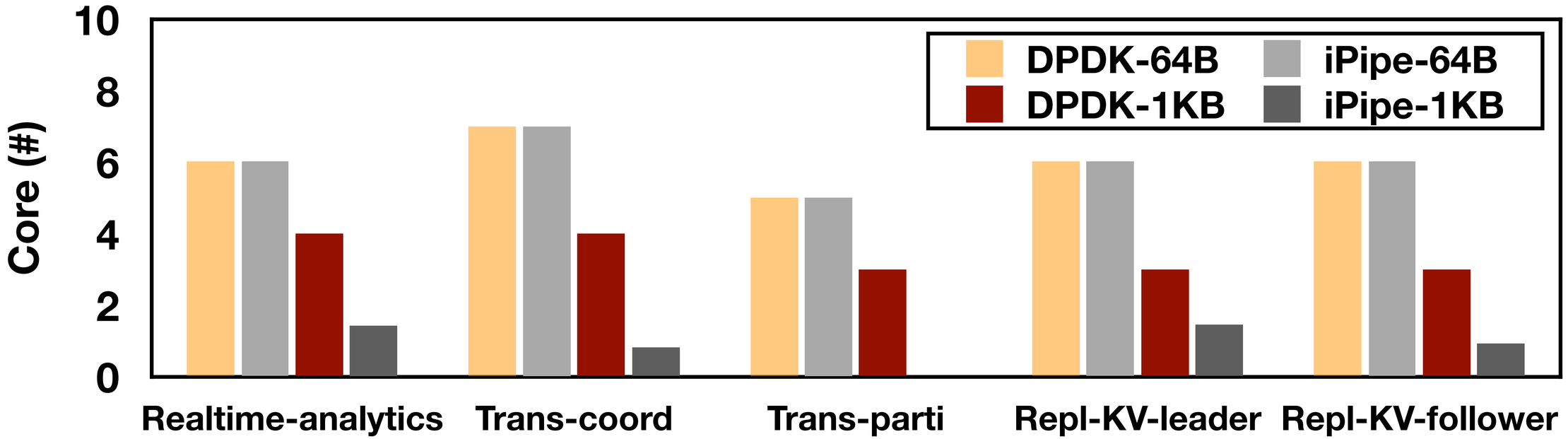
- Testbed
  - Supermicro servers, 12-core E5-2680 v3 Xeon CPUs



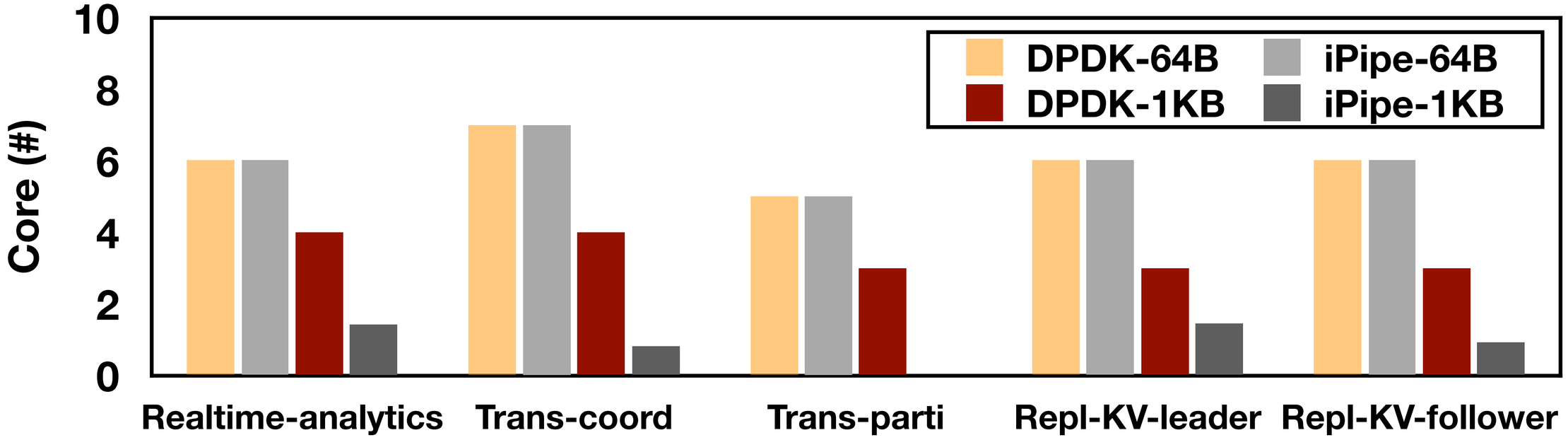
Host core savings for 25Gbps LiquidIOII CN2360



Host core savings for 25Gbps LiquidIOII CN2360



Host core savings for 25Gbps LiquidIOII CN2360



- Offloading adapts to traffic workload
- Average reduction in host core count is 73% for 1KB packets

Why SmartNICs?

#1: SmartNICs can reduce the request latency.
#2: SmartNICs can improve the throughput.
#3: SmartNICs can save host core cycles.
#4: SmartNICs can improve energy efficiency.
#5: SmartNICs can reduce the traffic load.
...
...

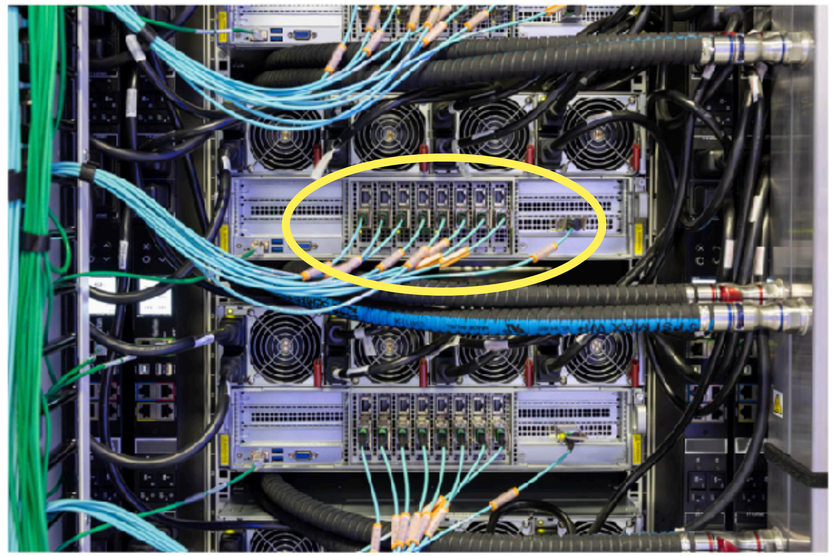
Benefits under Contingency

#1: SmartNICs can reduce the request latency if…
#2: SmartNICs can improve the throughput if…
#3: SmartNICs can save host core cycles if…
#4: SmartNICs can improve energy efficiency if…
#5: SmartNICs can reduce the traffic load if…
…
…

SmartNICs in the Real World

- Case #1: AWS nitro card
  - Annapurna Labs acquired in 2015
  - Packet processing and I/O data-plane offloading

- Case #2: Colossus cluster from xAI
  - 100K NVIDIA H100 GPUs
  - 400GbE Ethernet
  - BlueField-3 SuperNIC + Spectrum-X
  - One SmartNIC One GPU



Summary

- Today
  - SmartNICs

- Next topic: Data Center Transport
  - DCTCP (Sigcomm'10)
  - NDP (Sigcomm'17)
  - Homa (Sigcomm'18)