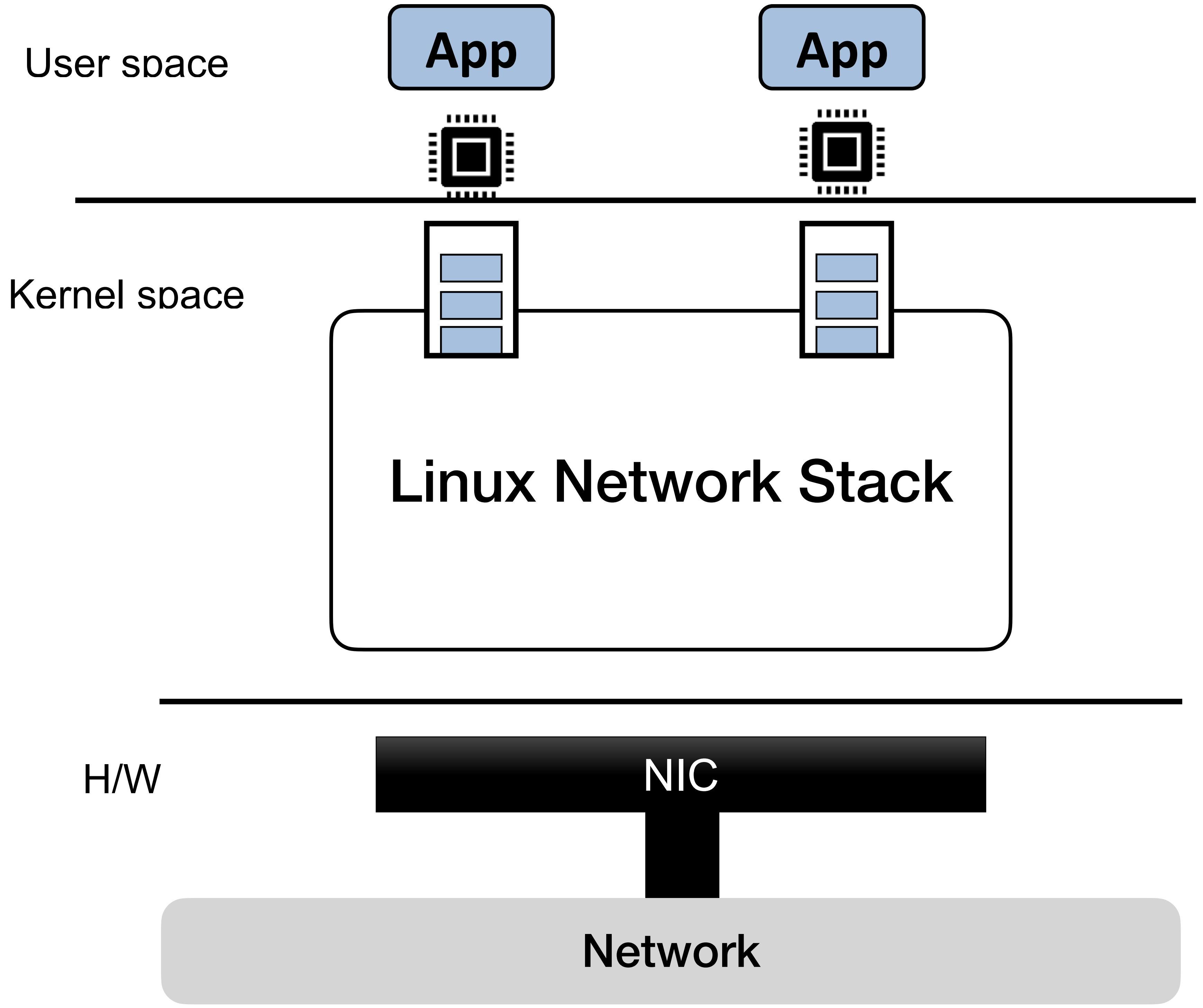
Towards μs Tail Latency and Terabit Ethernet: Disaggregating the Host Network Stack

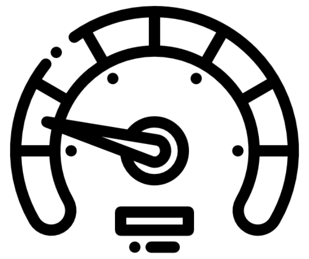




Qizhe Cai | Midhul Vuppalapati | Jaehyun Hwang | Christos Kozyrakis | Rachit Agarwal

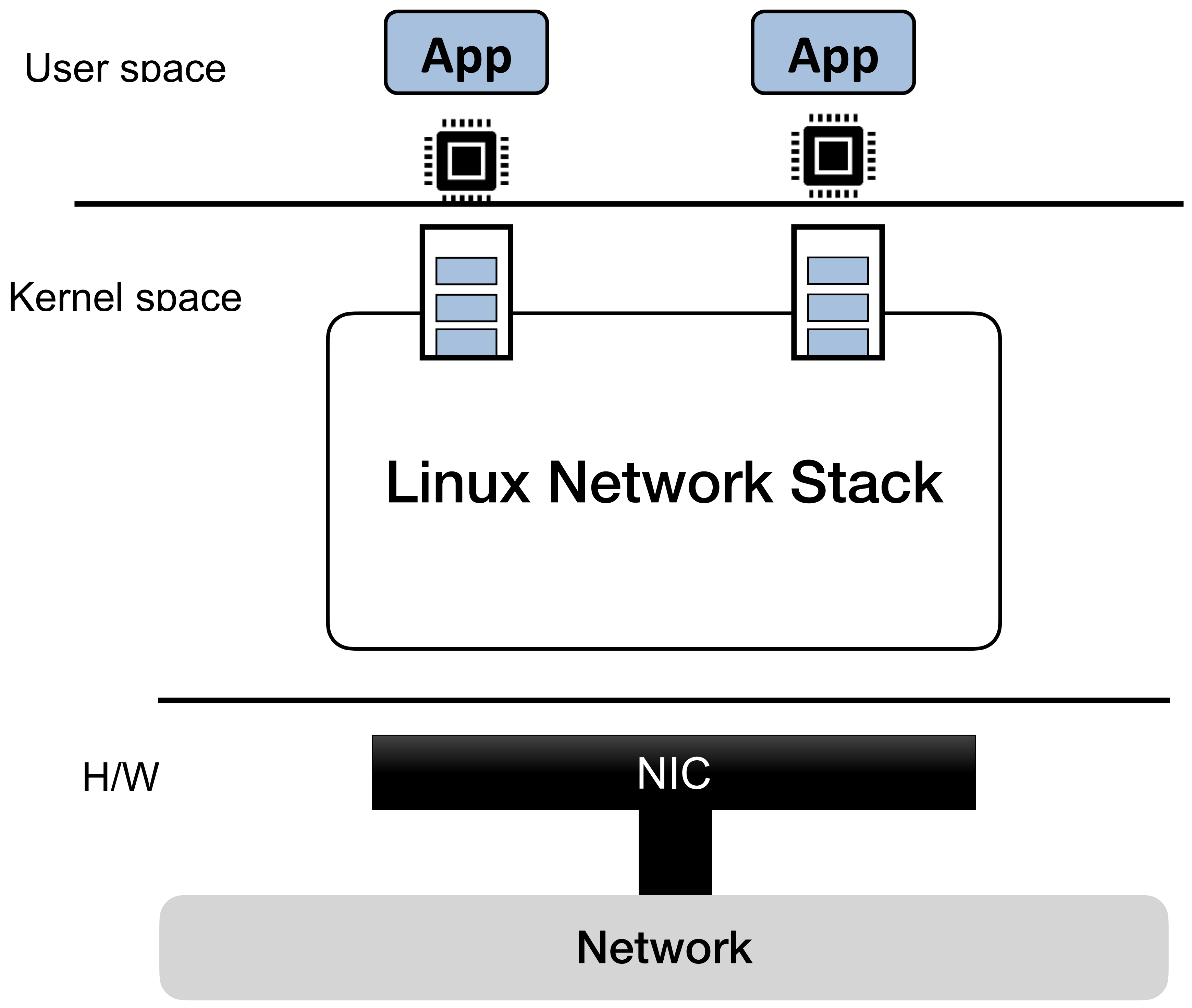
Limitations of Existing Host Network Stacks



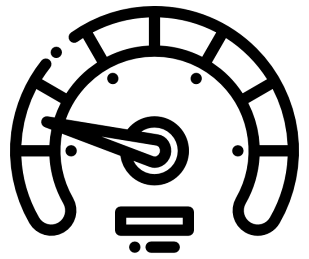


Inefficient Packet Processing Pipeline

Limitations of Existing Host Network Stacks

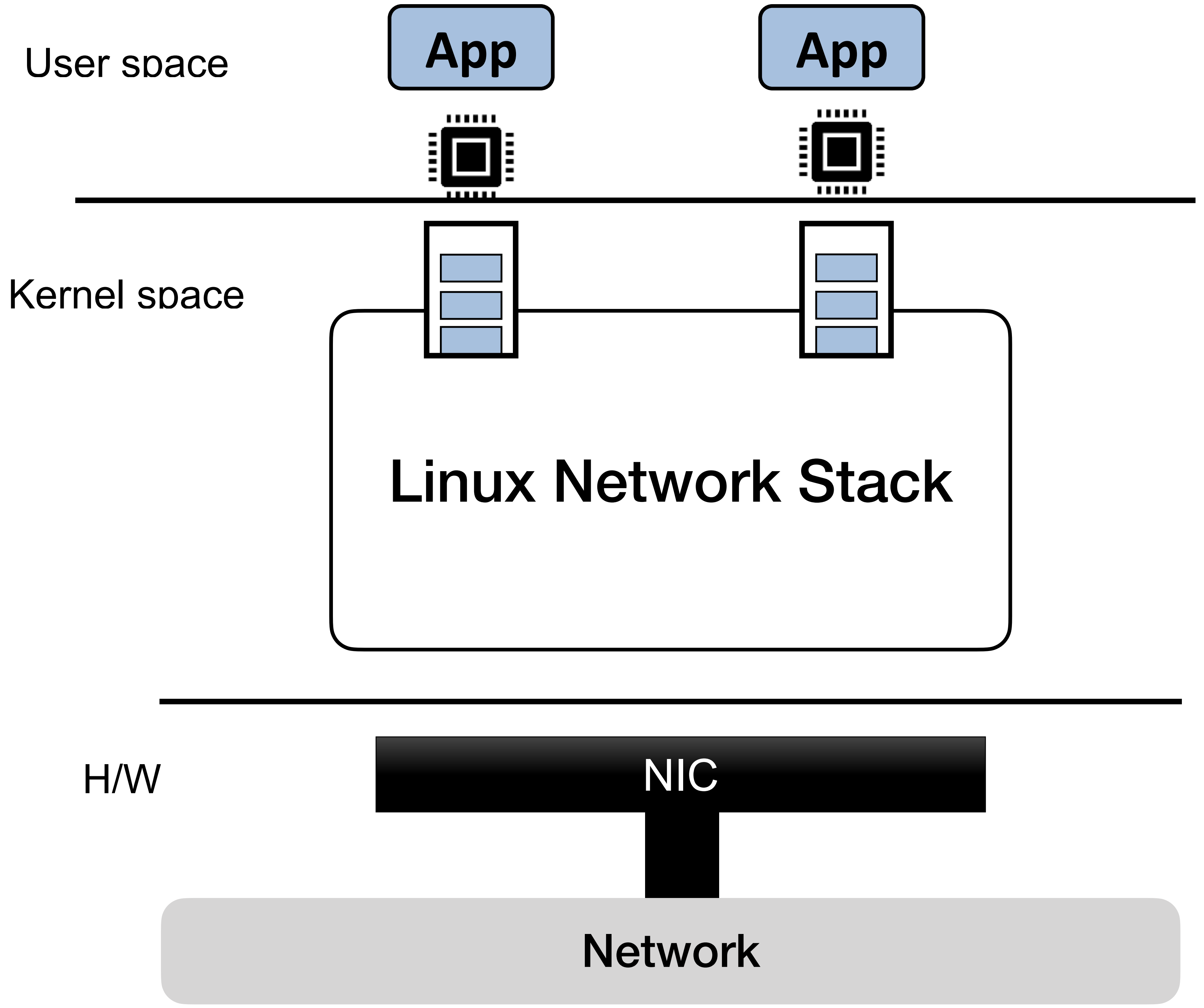


Frequently Cited Problems

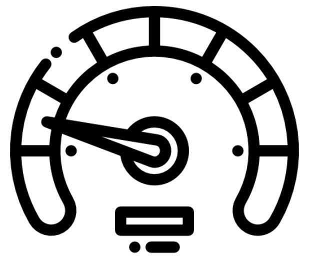


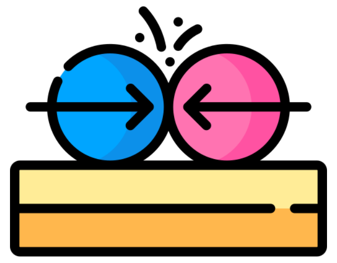
Inefficient Packet Processing Pipeline

Limitations of Existing Host Network Stacks

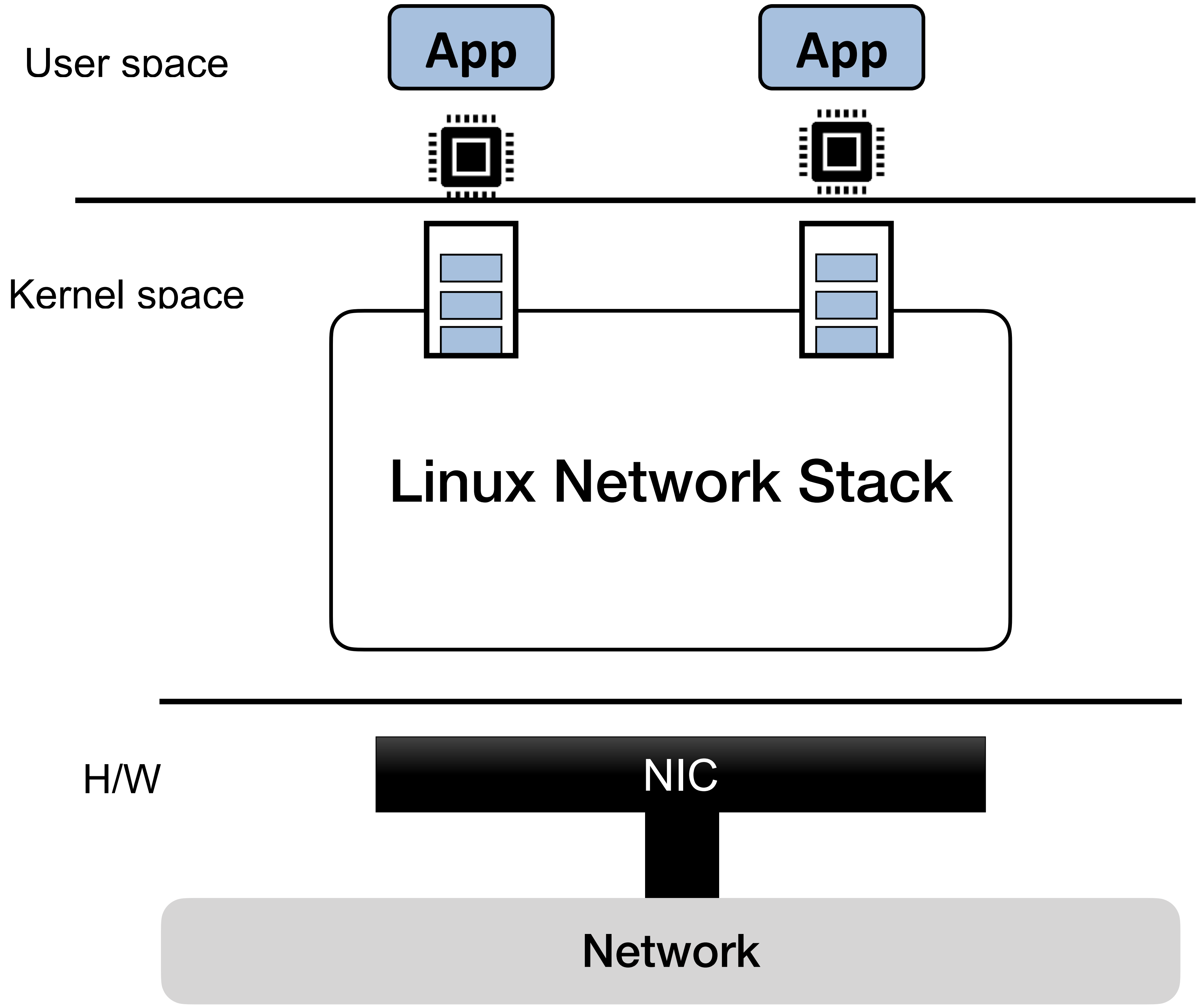


Frequently Cited Problems

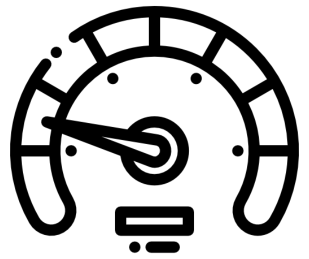

Inefficient Packet Processing Pipeline

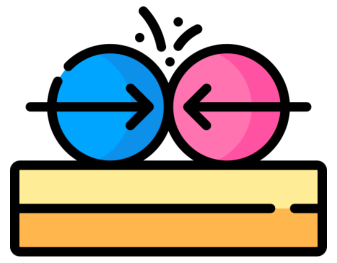

Poor Performance Isolation

Limitations of Existing Host Network Stacks



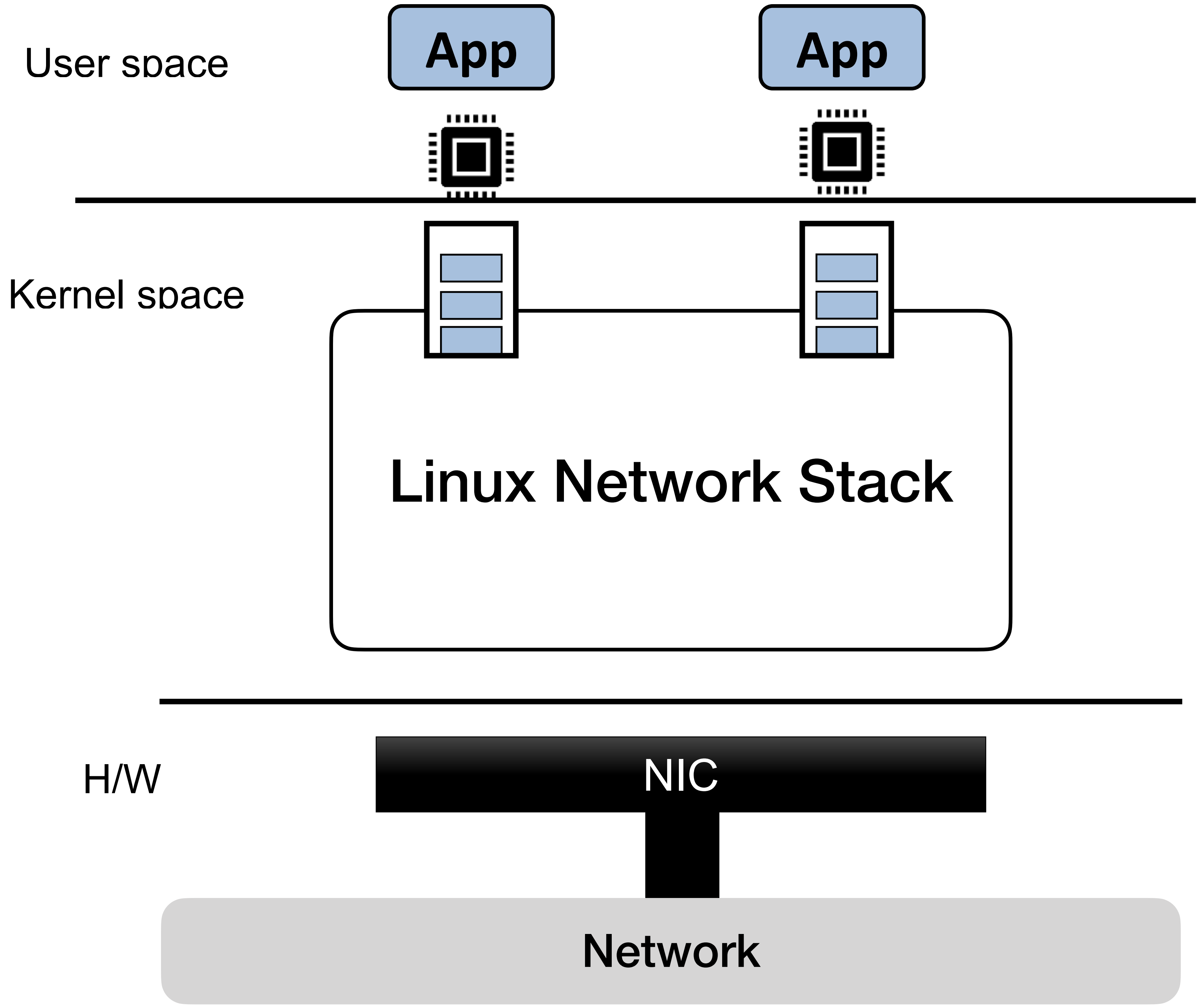
Frequently Cited Problems


Inefficient Packet Processing Pipeline

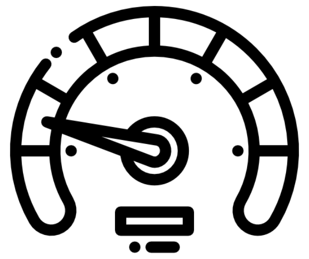

Poor Performance Isolation

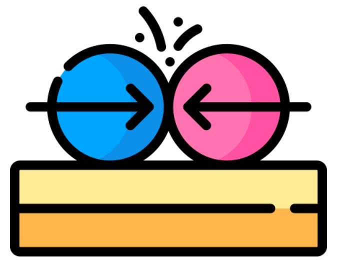

Rigid & Complex Implementation

Limitations of Existing Host Network Stacks

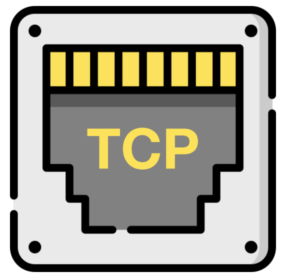


Frequently Cited Problems


Inefficient Packet Processing Pipeline


Poor Performance Isolation

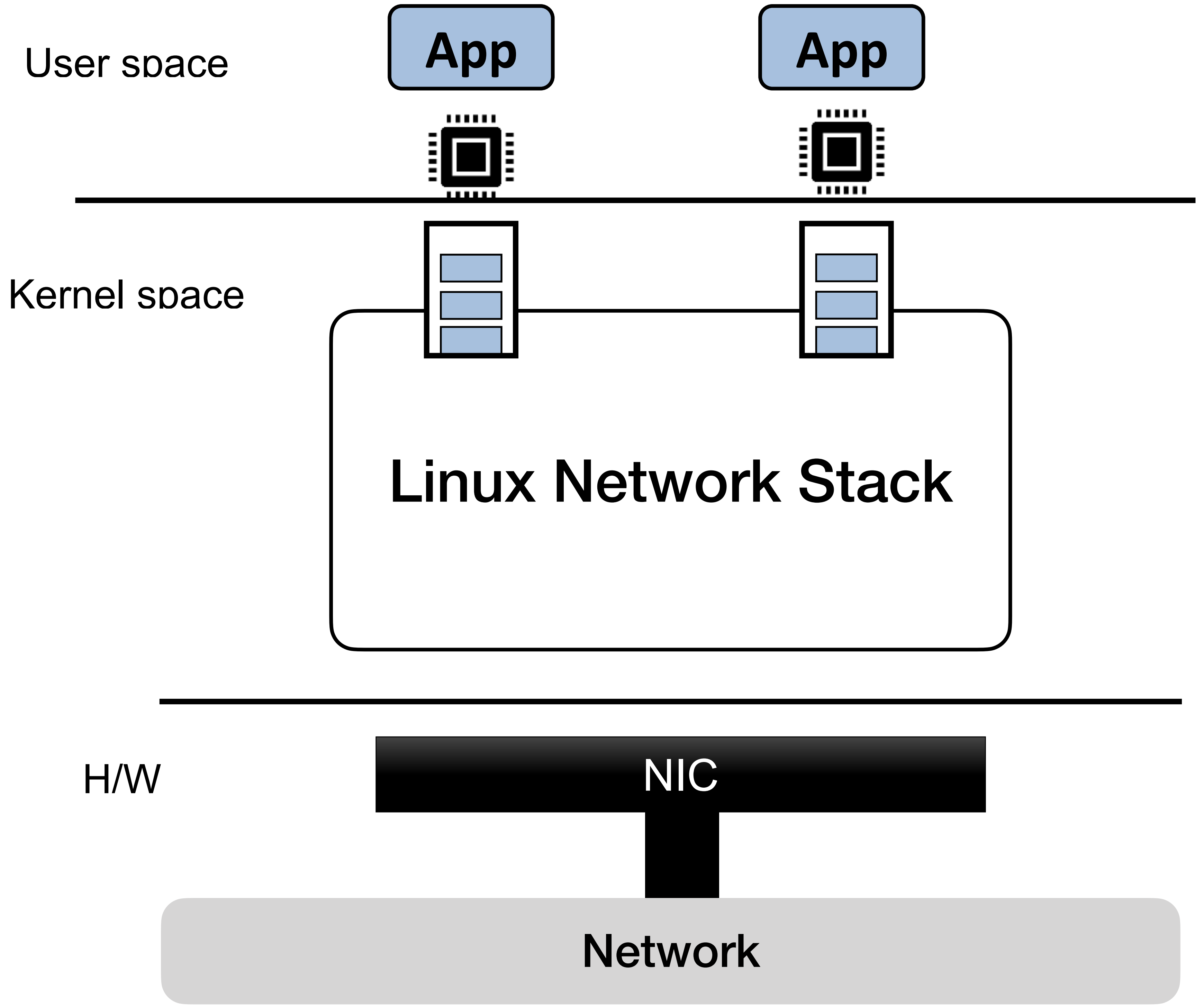
Rigid & Complex Implementation



Inefficient Transport Protocols

.
.
.

Limitations of Existing Host Network Stacks



Frequently Cited Problems

- Inefficient Packet Processing Pipeline
- Poor Performance Isolation
- Rigid & Complex Implementation
- Inefficient Transport Protocols

.
.
.

Many interesting debates on various design aspects

Debates on various design aspects

Debates on various design aspects

Interface

Streaming

RPC

Debates on various design aspects

Interface

- Streaming
- RPC

Semantics

- Synchronous
- Asynchronous

Debates on various design aspects

Interface
- Streaming
- RPC

Semantics
- Synchronous
- Asynchronous

Placement
- In-kernel
- Userspace
- Hardware

Debates on various design aspects

Interface
- Streaming
- RPC

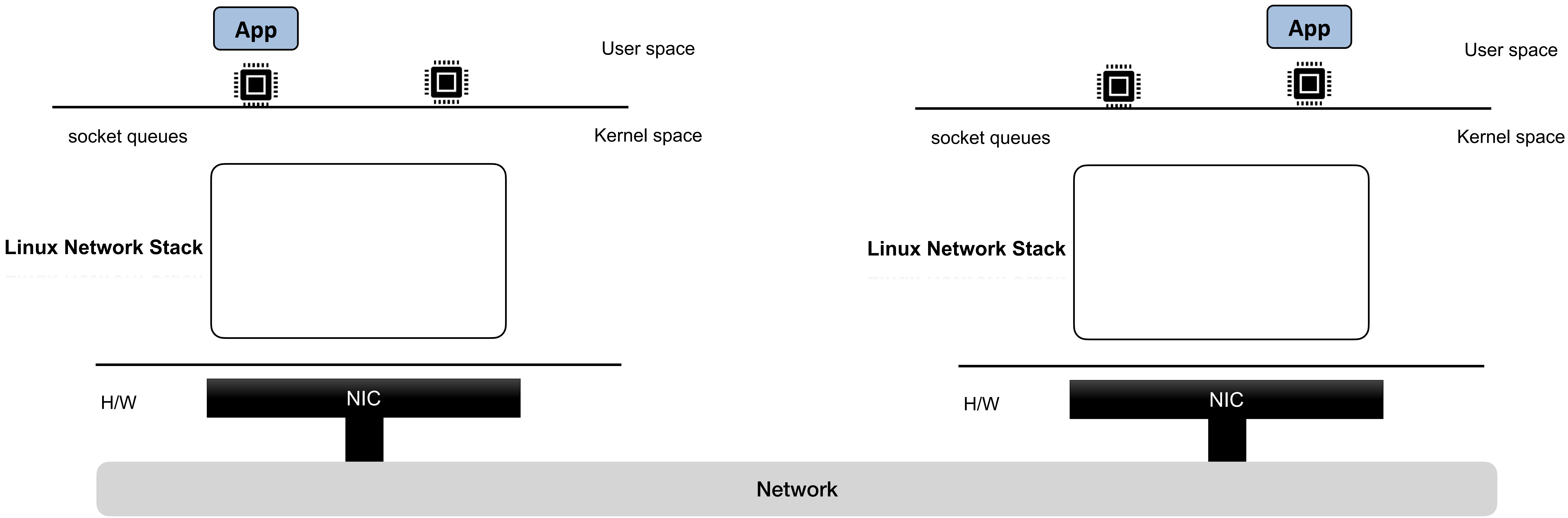
Semantics
- Synchronous
- Asynchronous

Placement
- In-kernel
- Userspace
- Hardware

This paper: many limitations of the Host Network Stack are *not* rooted in Interface, Semantics or Placement but rather in its Core Architecture

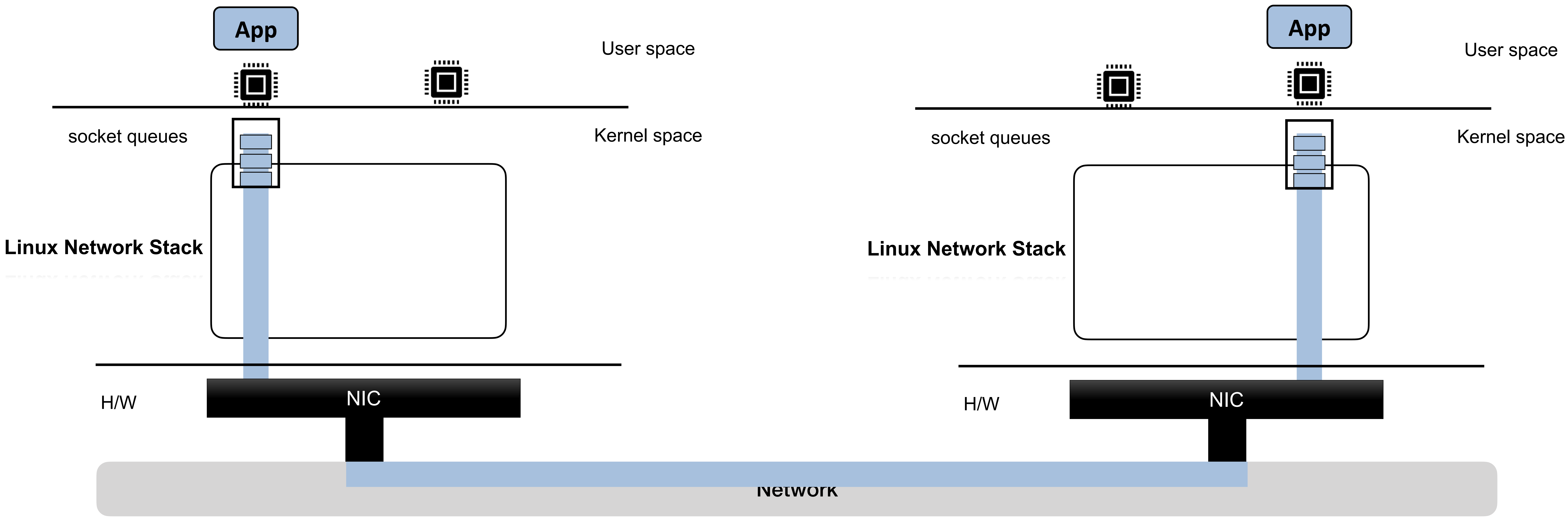
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a **“pipe” abstraction**



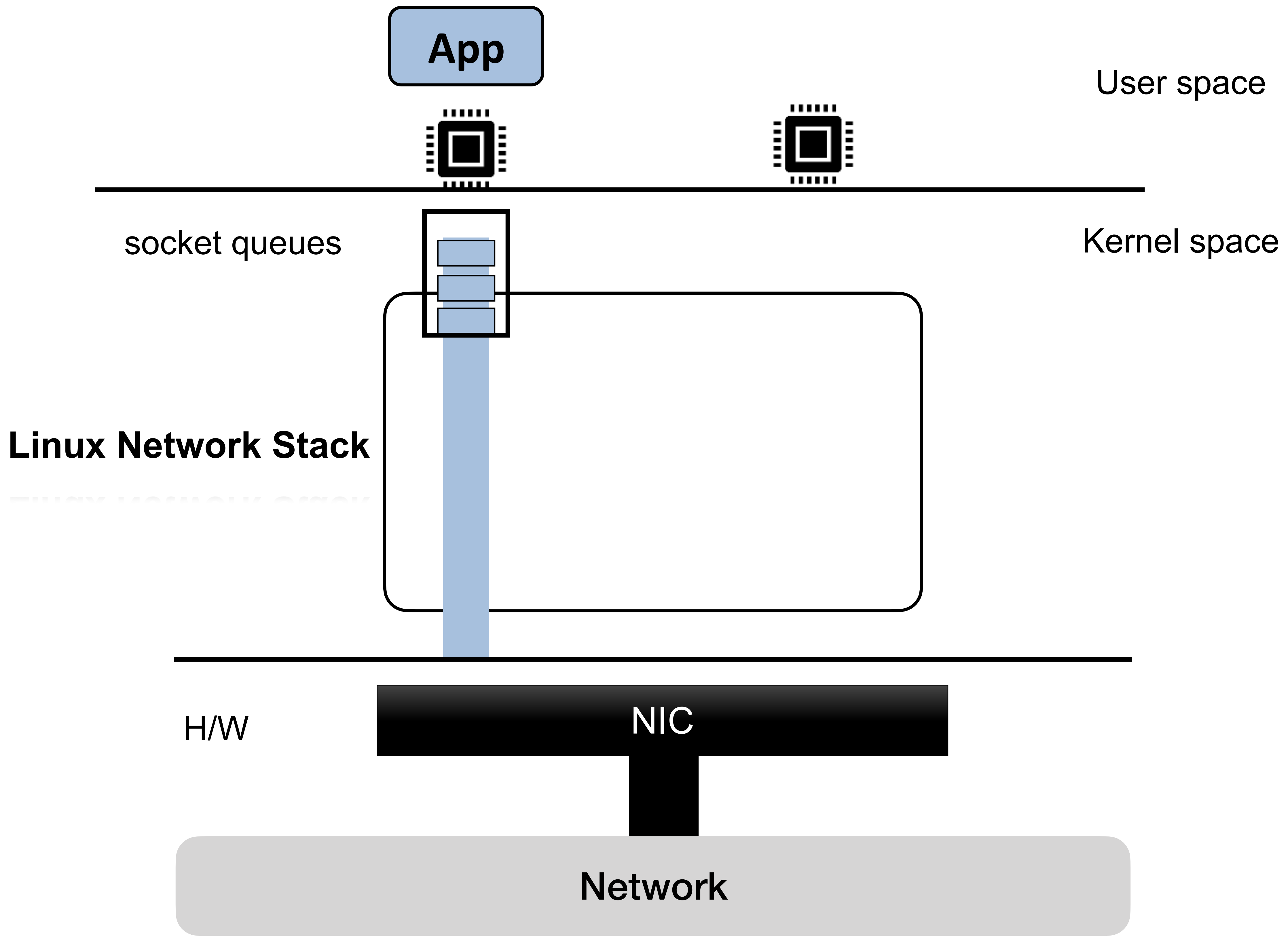
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a **“pipe” abstraction**



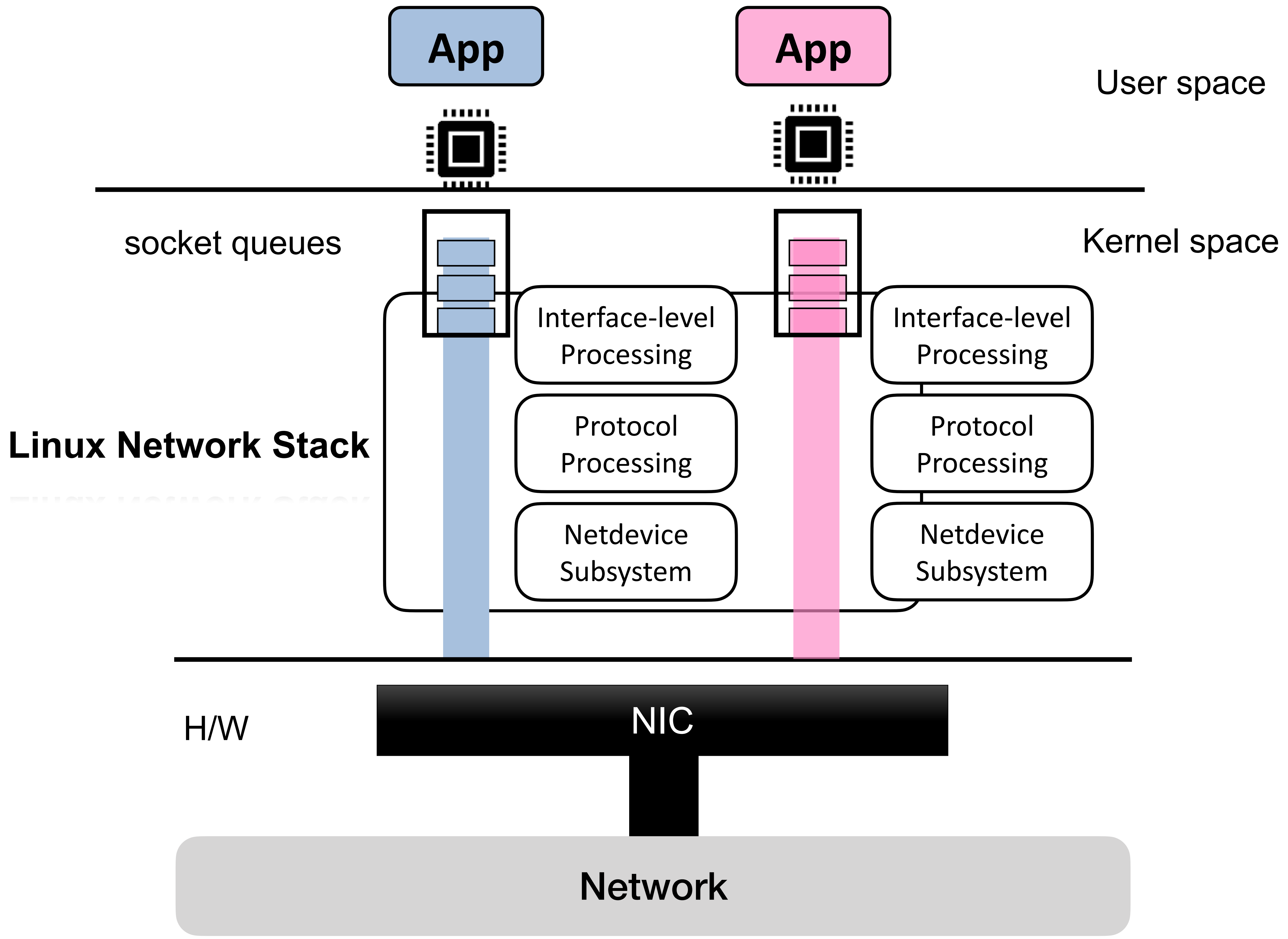
Architecture of Existing Host Network Stacks

Existing Stacks offer applications a **"pipe" abstraction**



Architecture of Existing Host Network Stacks

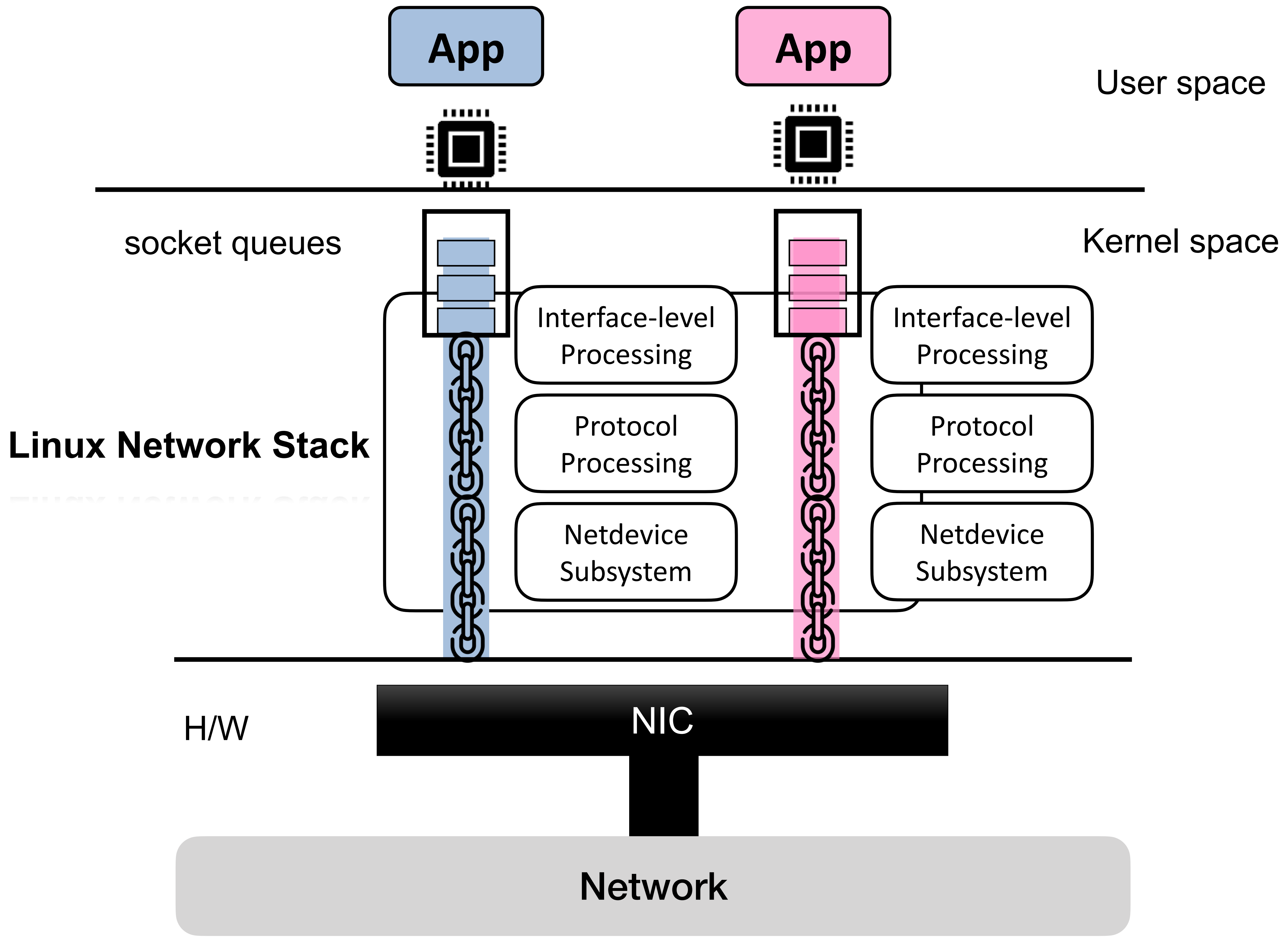
Existing Stacks offer applications a **“pipe” abstraction**



Dedicated
Each application/socket has an independent instance of packet processing pipeline

Architecture of Existing Host Network Stacks

Existing Stacks offer applications a **“pipe” abstraction**

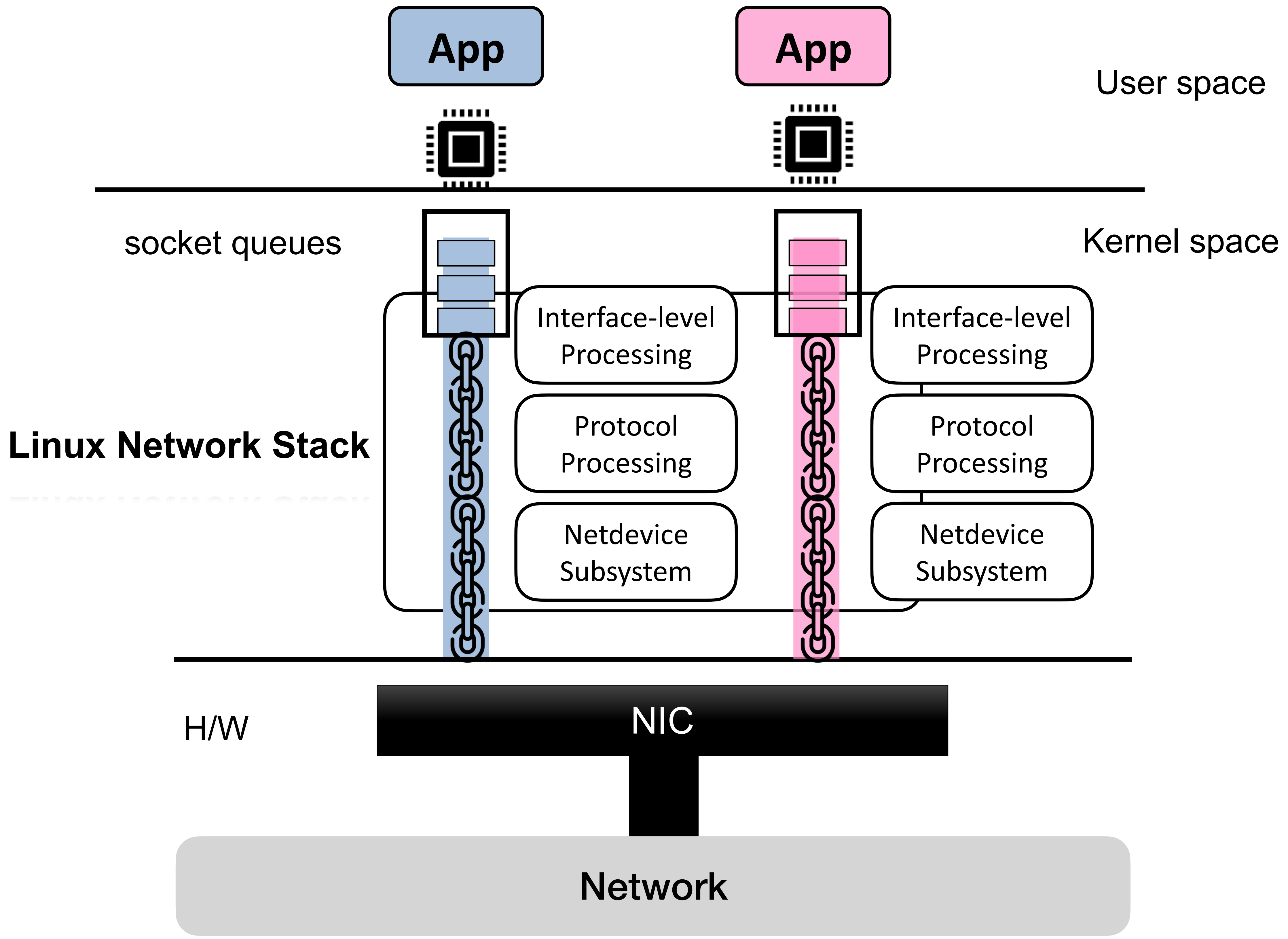


Dedicated
Each application/socket has an independent instance of packet processing pipeline

Tightly Integrated
Packet processing coupled to application cores

Architecture of Existing Host Network Stacks

Existing Stacks offer applications a **“pipe” abstraction**



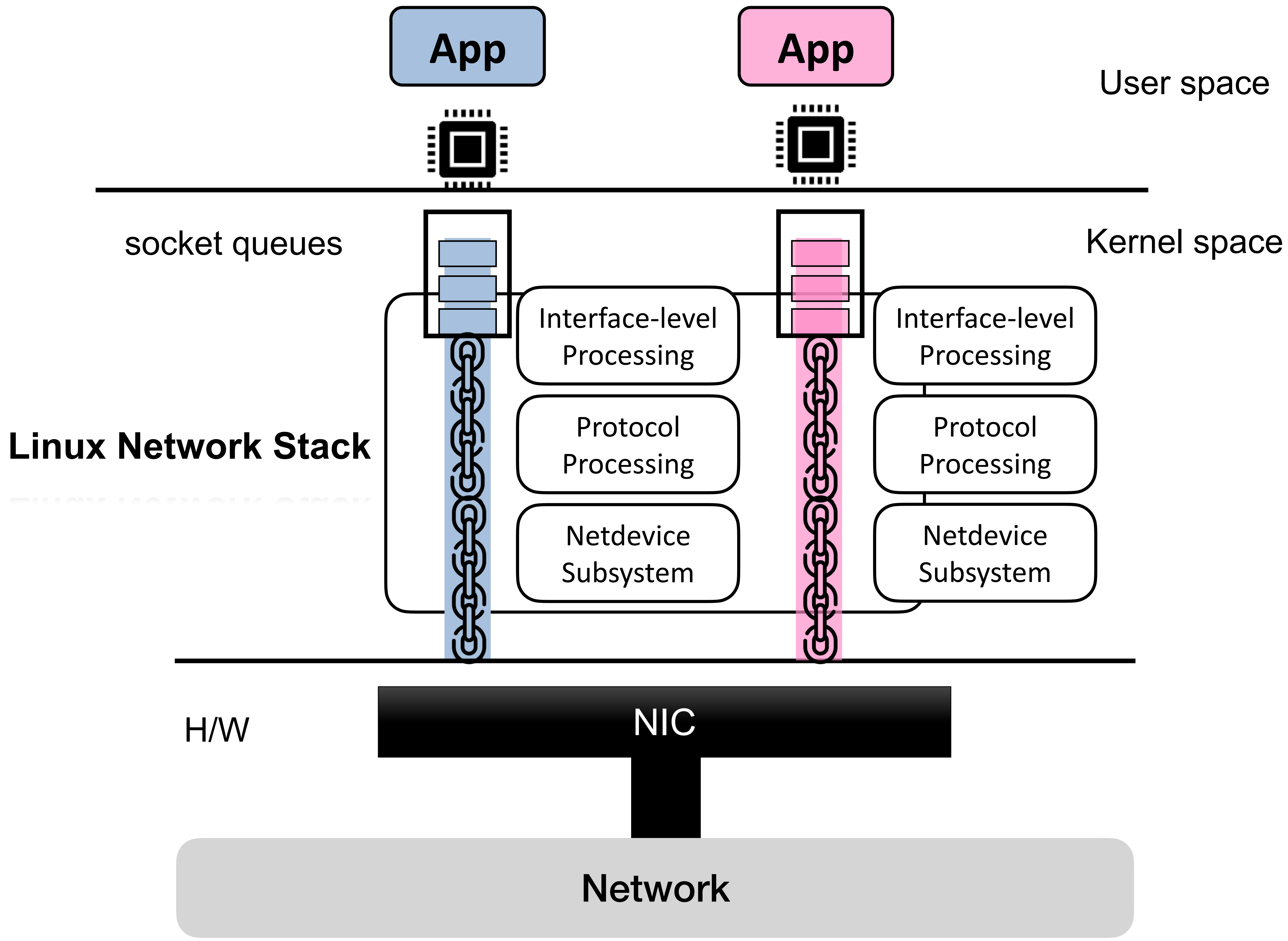
Dedicated
Each application/socket has an independent instance of packet processing pipeline

Tightly Integrated
Packet processing coupled to application cores

Static
Host resource provisioning determined at pipe creation (Independent of other pipes and resource availability)

Architecture of Existing Host Network Stacks

Existing Stacks offer applications a **“pipe” abstraction**



Dedicated
Each application/socket has an independent instance of packet processing pipeline

Tightly Integrated
Packet processing coupled to application cores

Static
Host resource provisioning determined at pipe creation (Independent of other pipes and resource availability)

Dedicated, Tightly Integrated, and Static Pipelines:
preclude network stacks from exploiting capabilities of modern hardware

This Work

This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation

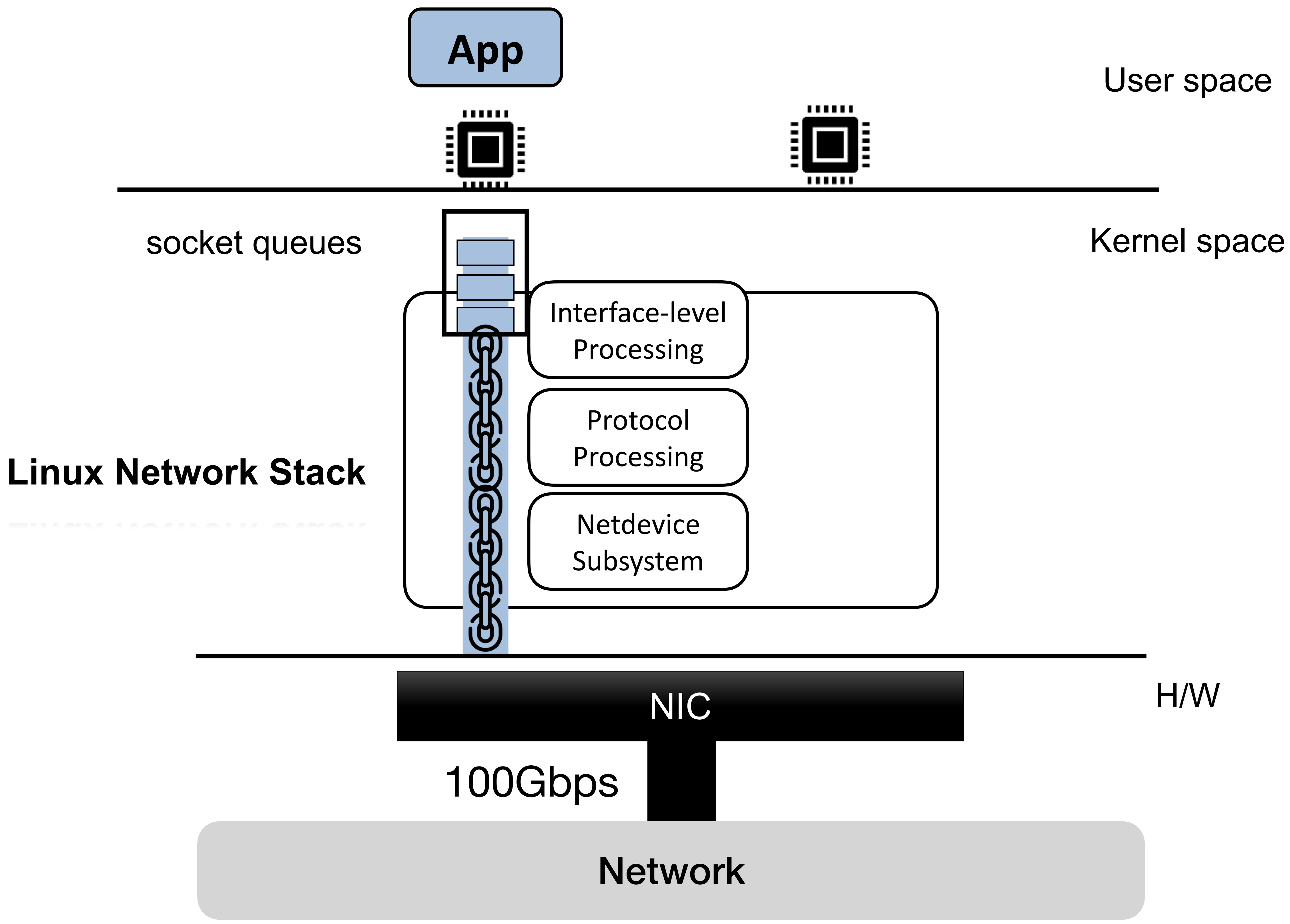
This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

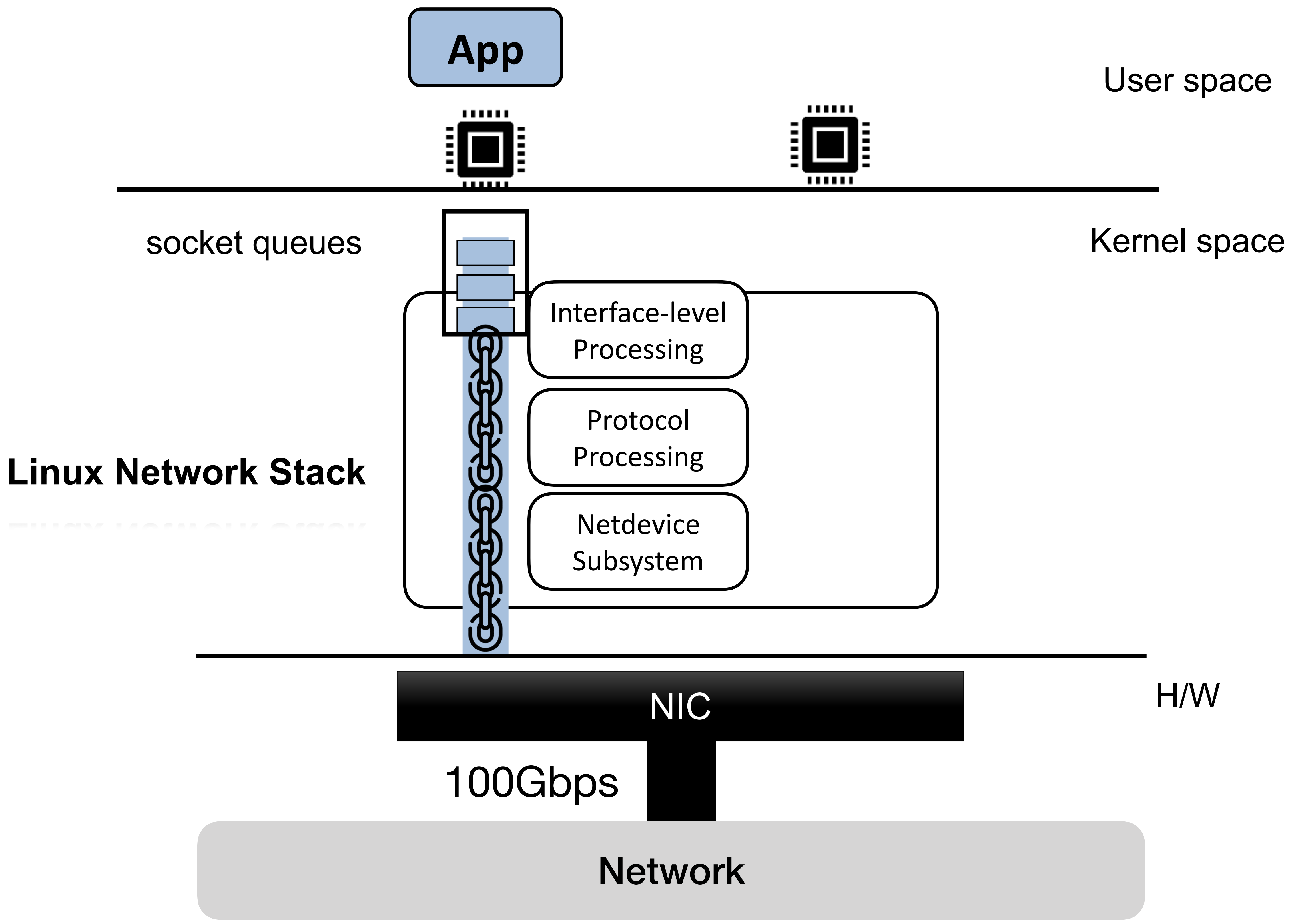
NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation

Problem with Static and Dedicated Pipes



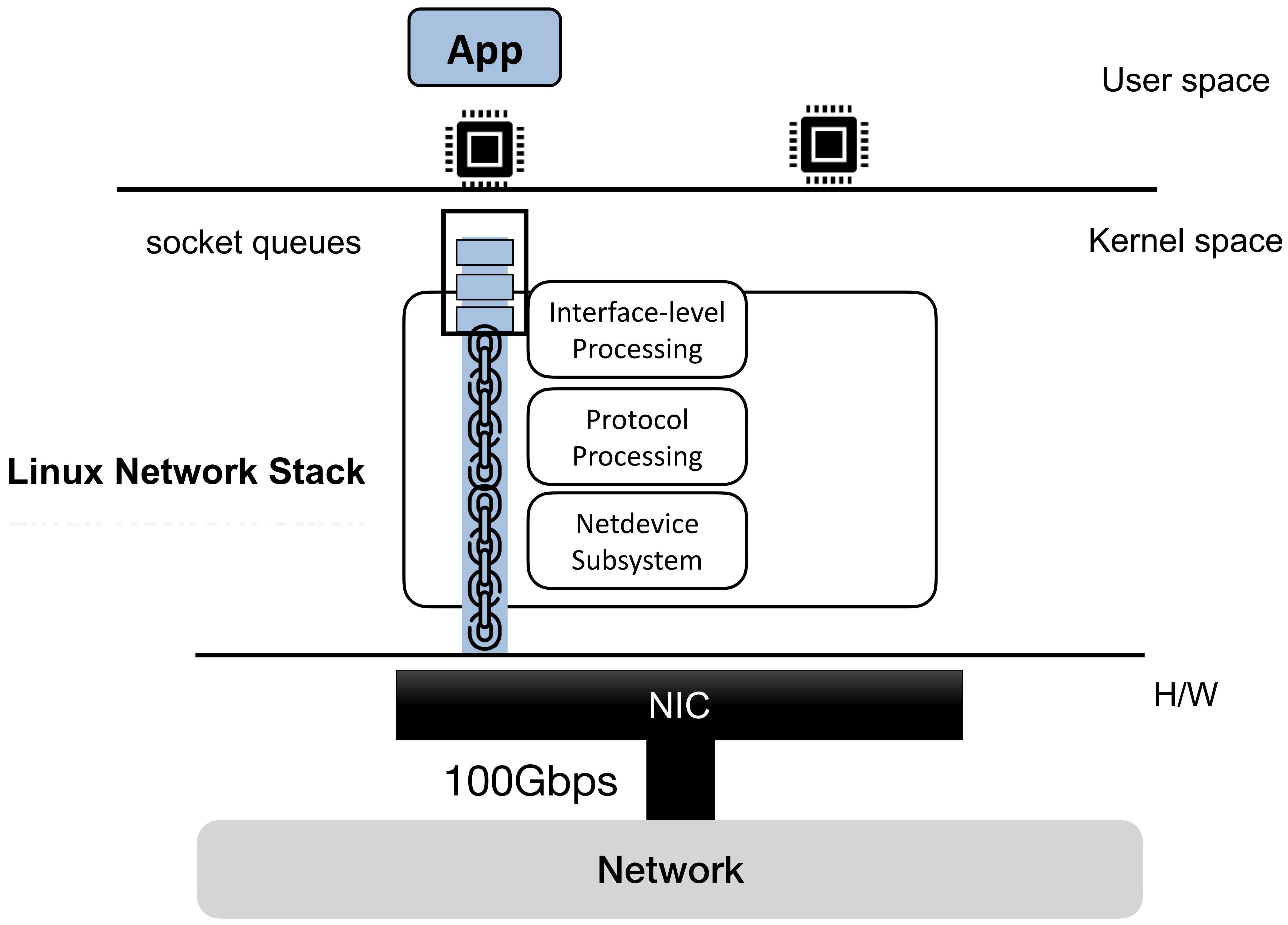
Problem with Static and Dedicated Pipes

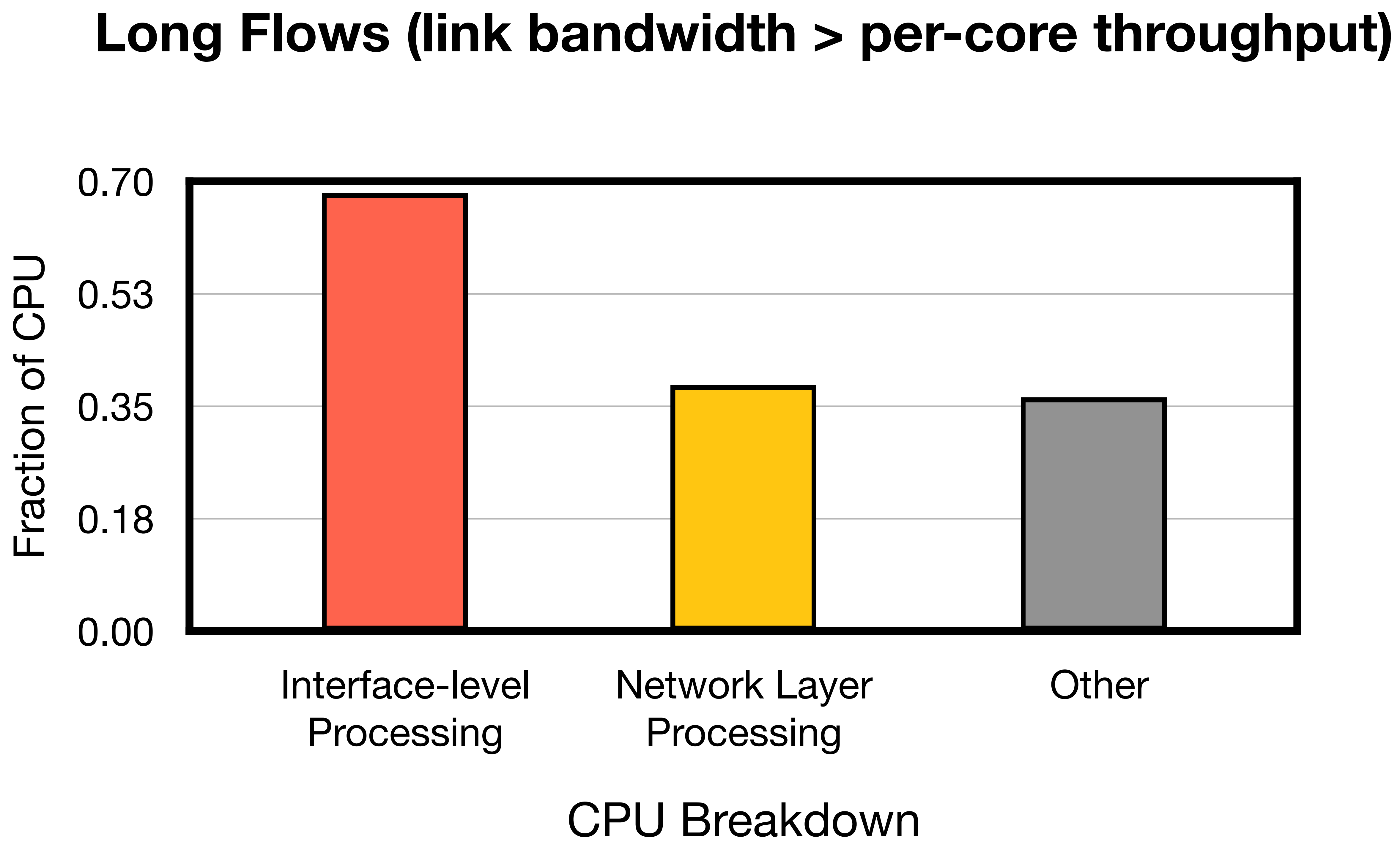


Long Flows (link bandwidth > per-core throughput)

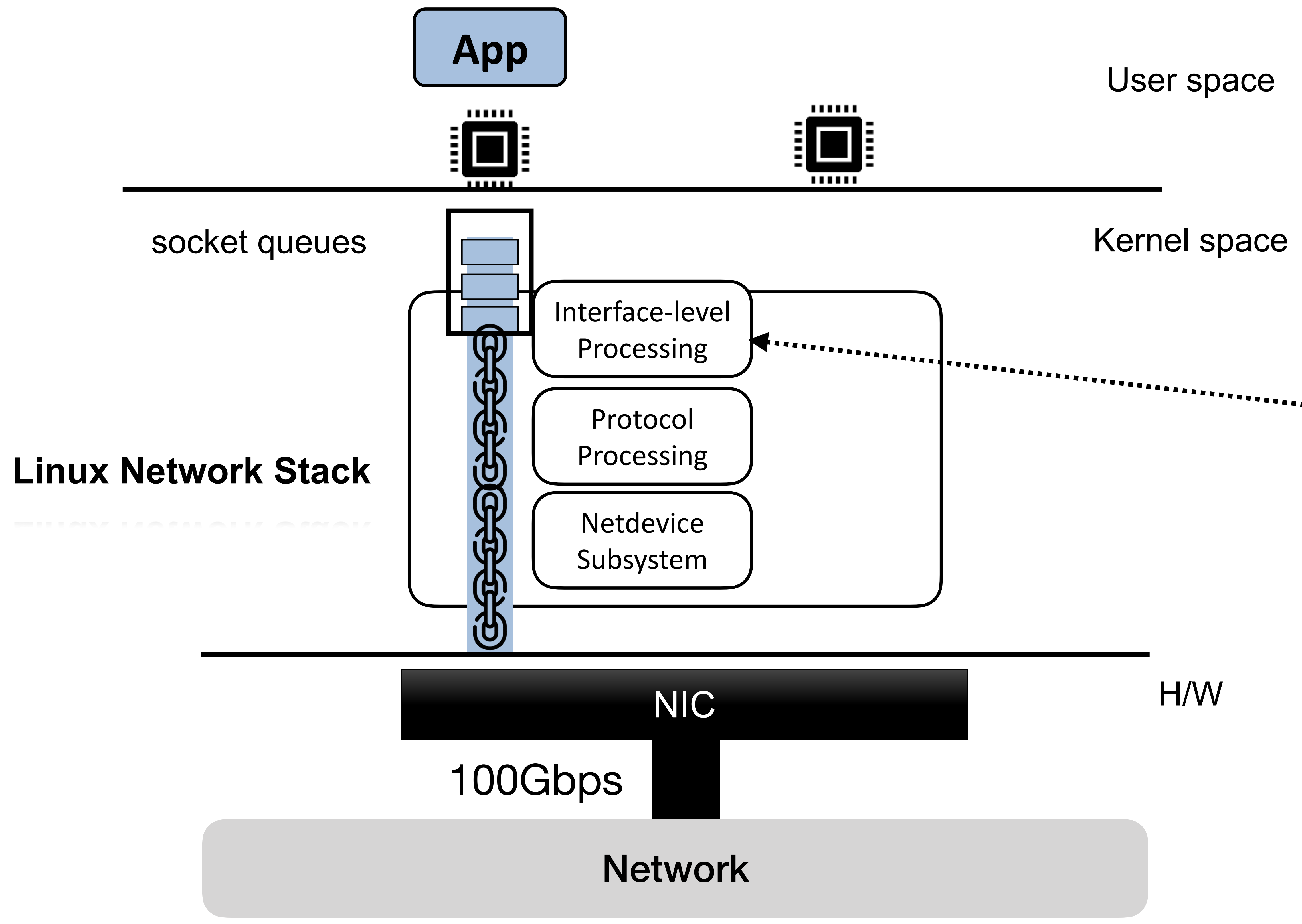
Icon made by Freepik from www.flaticon.com

Problem with Static and Dedicated Pipes

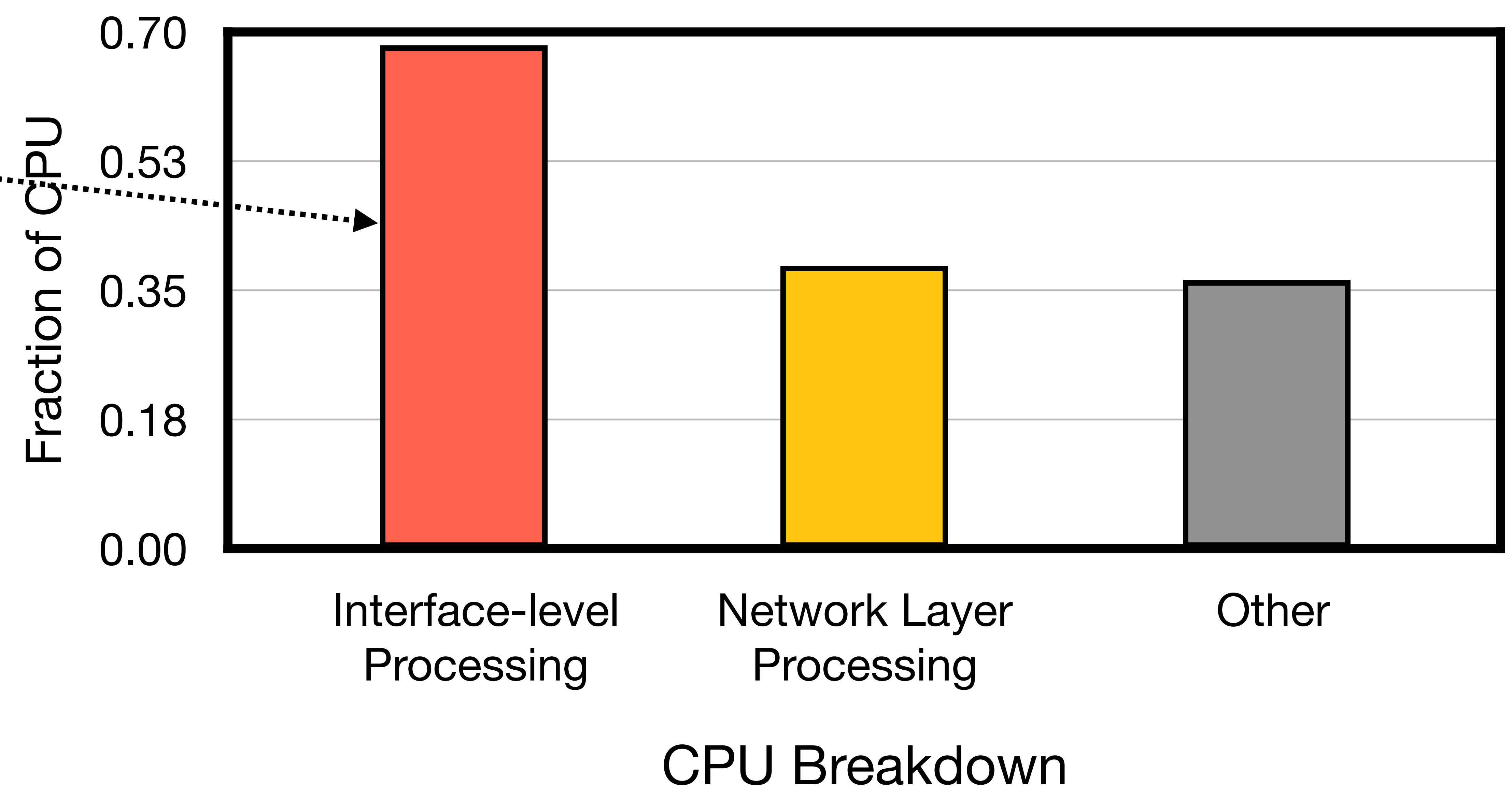




Problem with Static and Dedicated Pipes

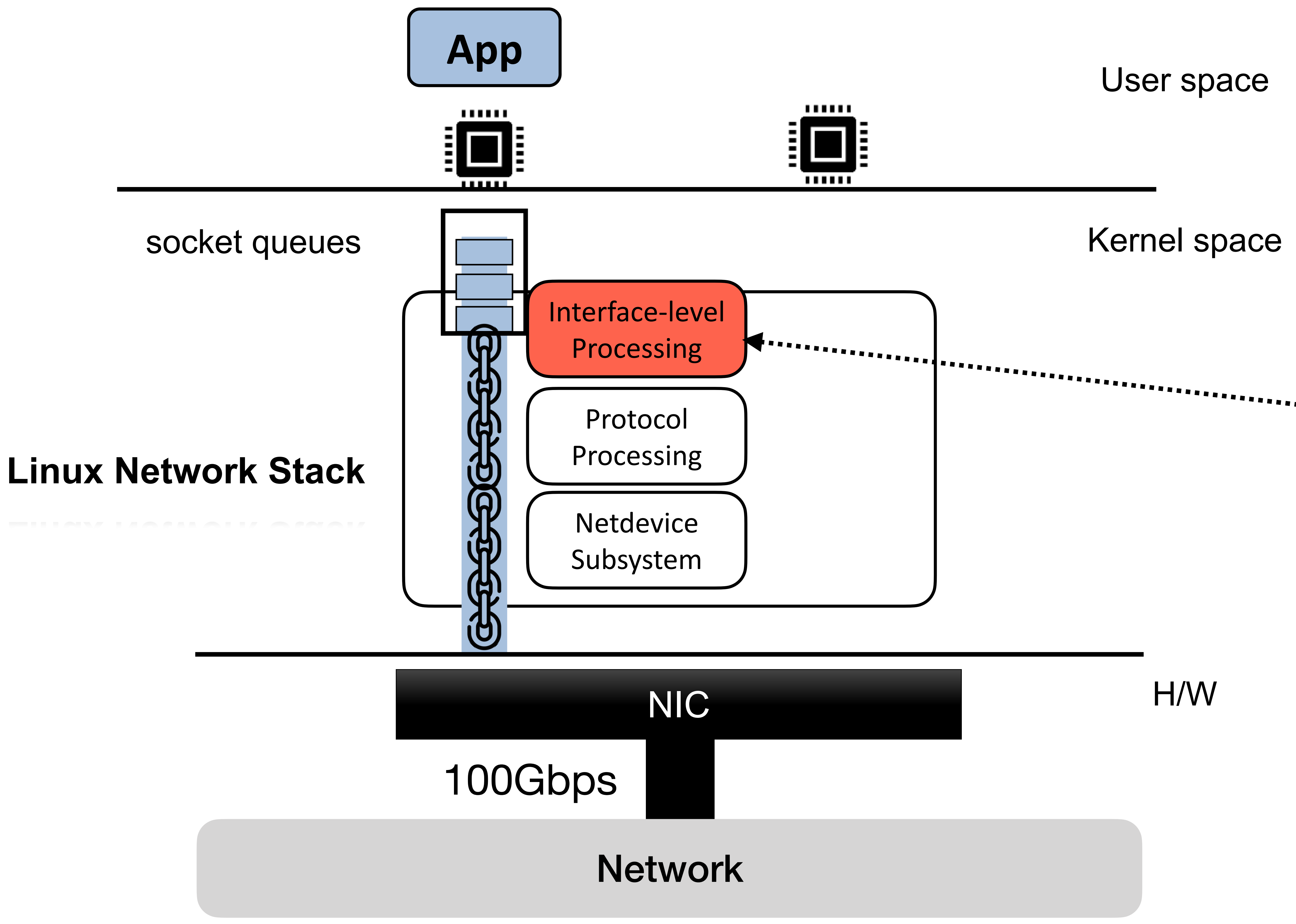


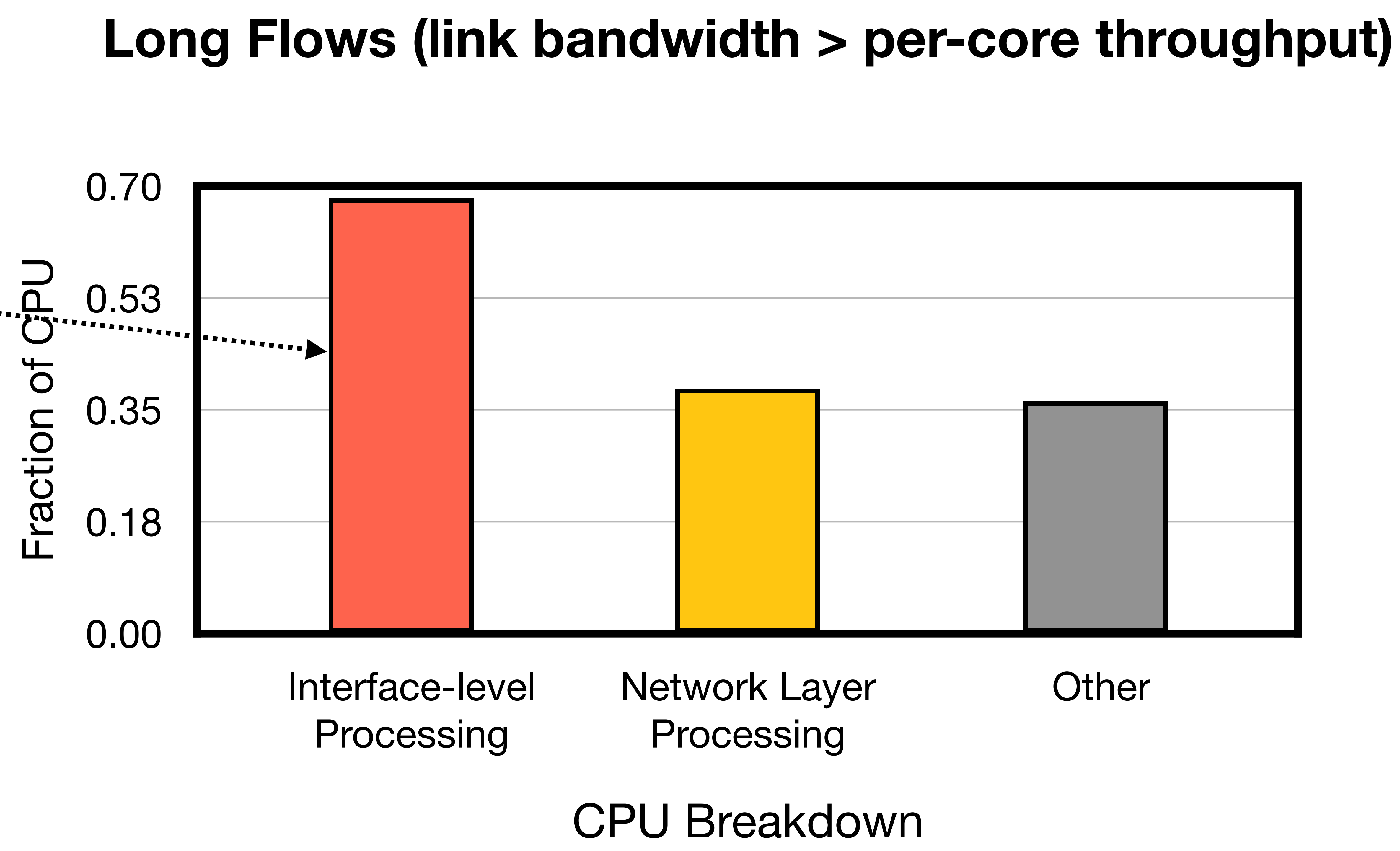
Long Flows (link bandwidth > per-core throughput)



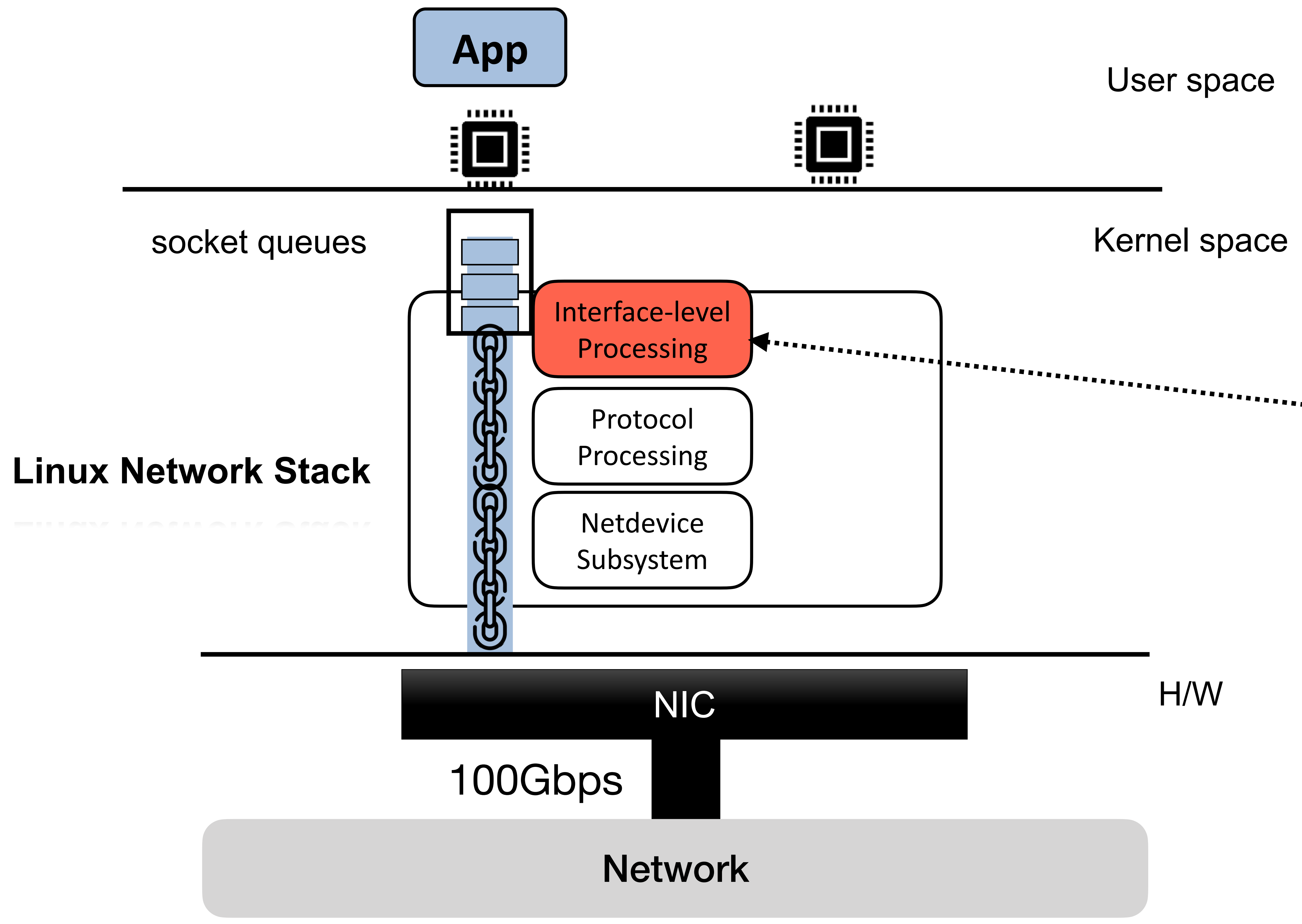
Icon made by Freepik from www.flaticon.com

Problem with Static and Dedicated Pipes

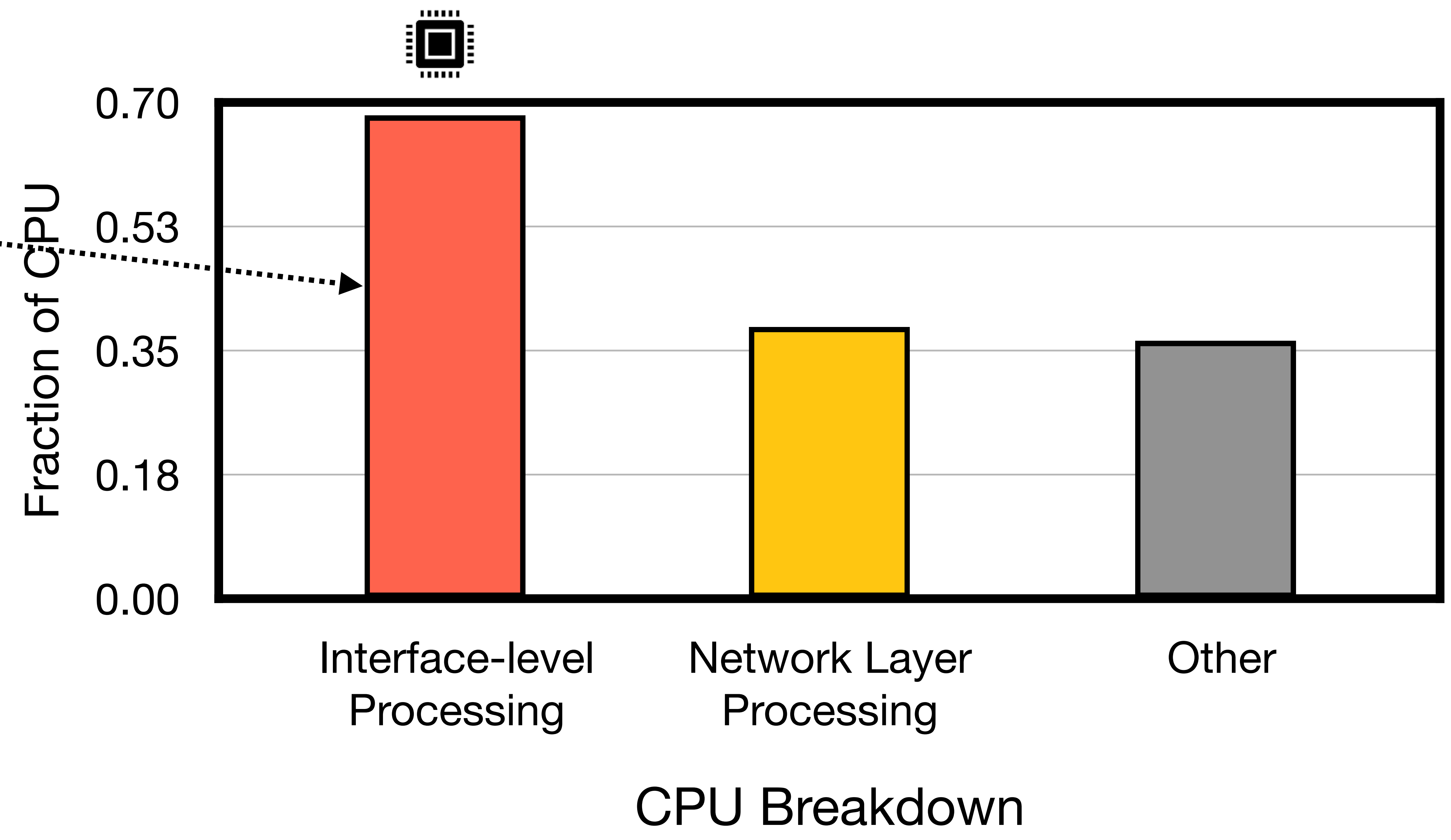




Problem with Static and Dedicated Pipes

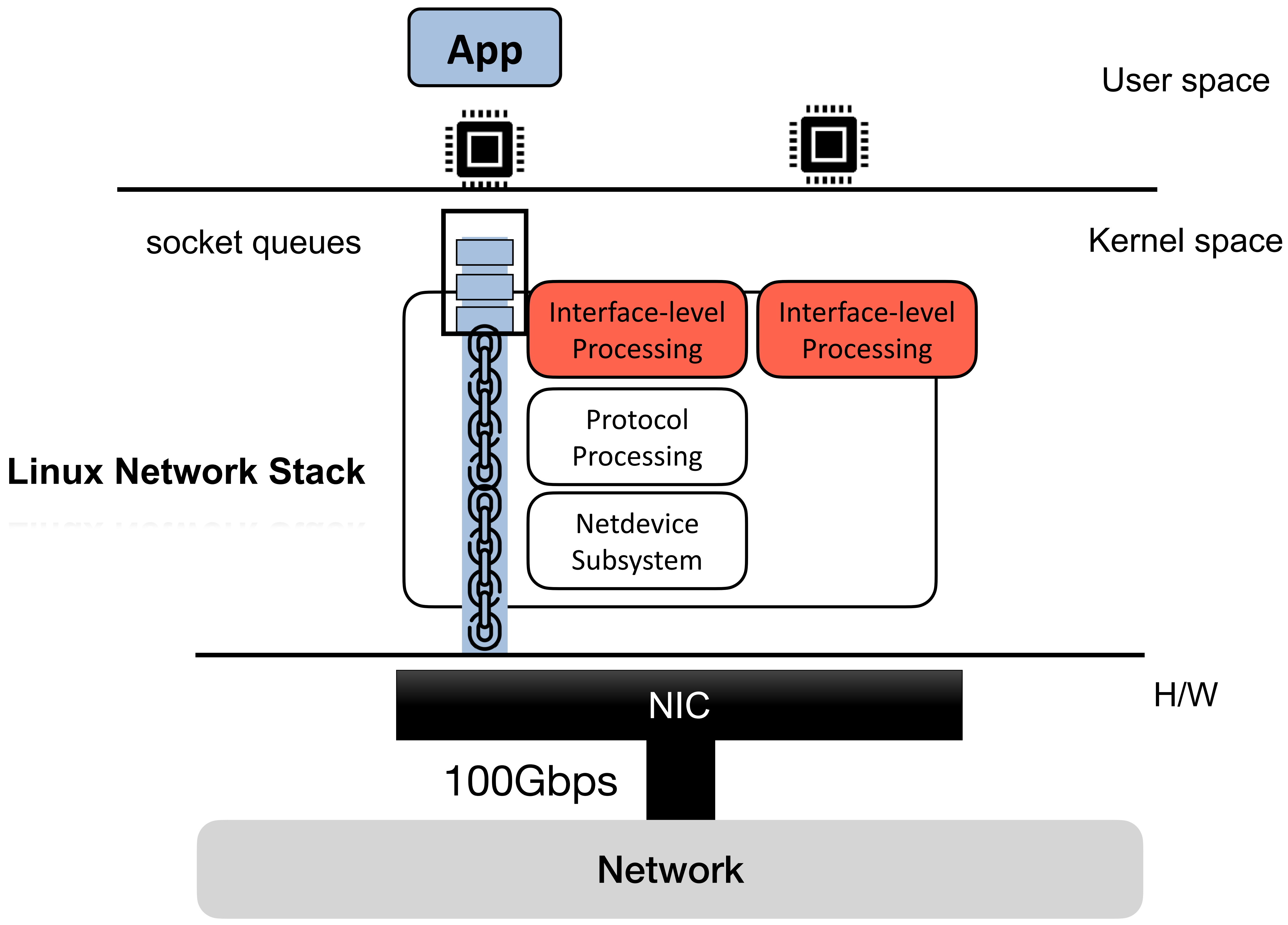


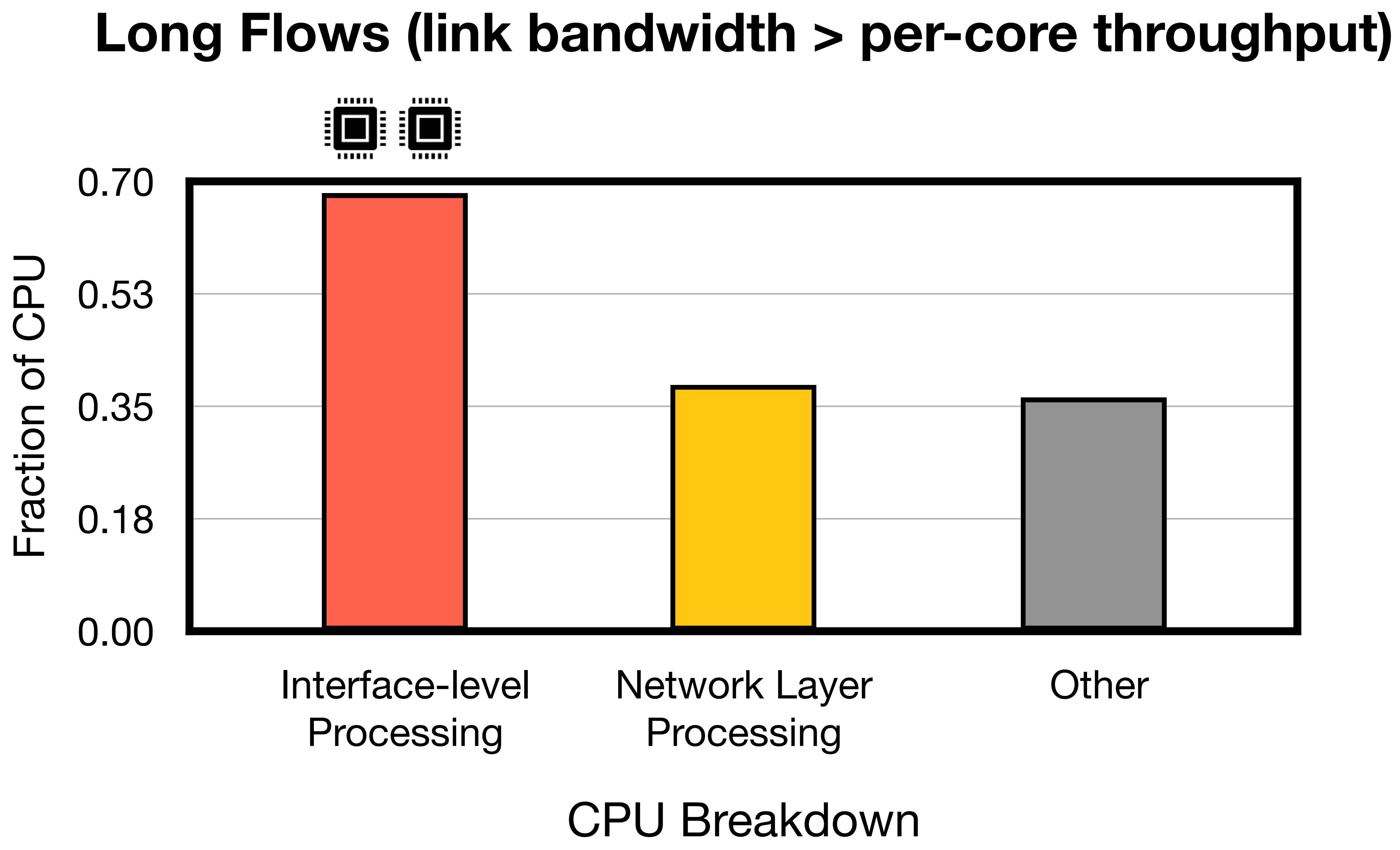
Long Flows (link bandwidth > per-core throughput)



Resources for Interface-level Processing limited by number of application cores

Problem with Static and Dedicated Pipes

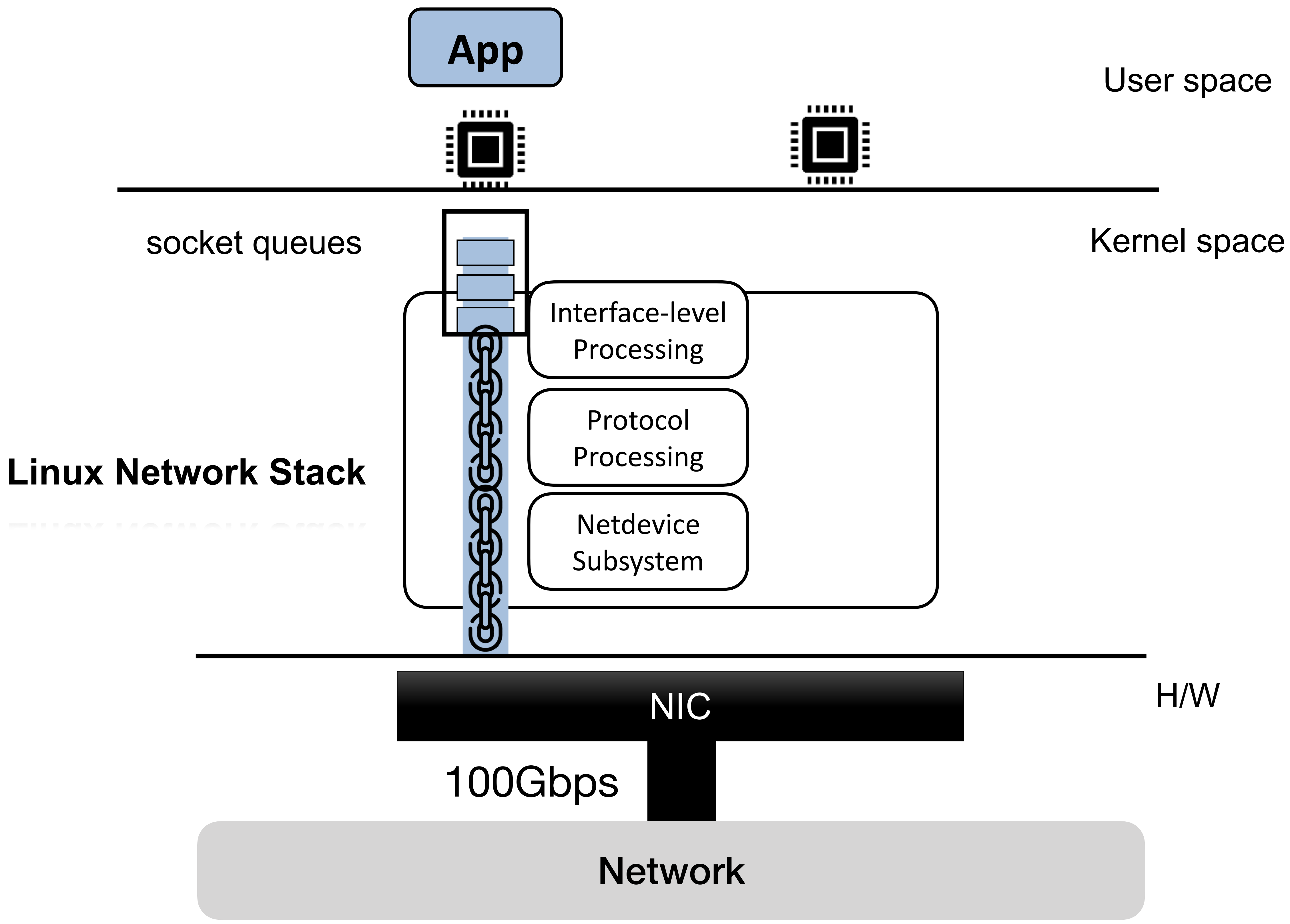




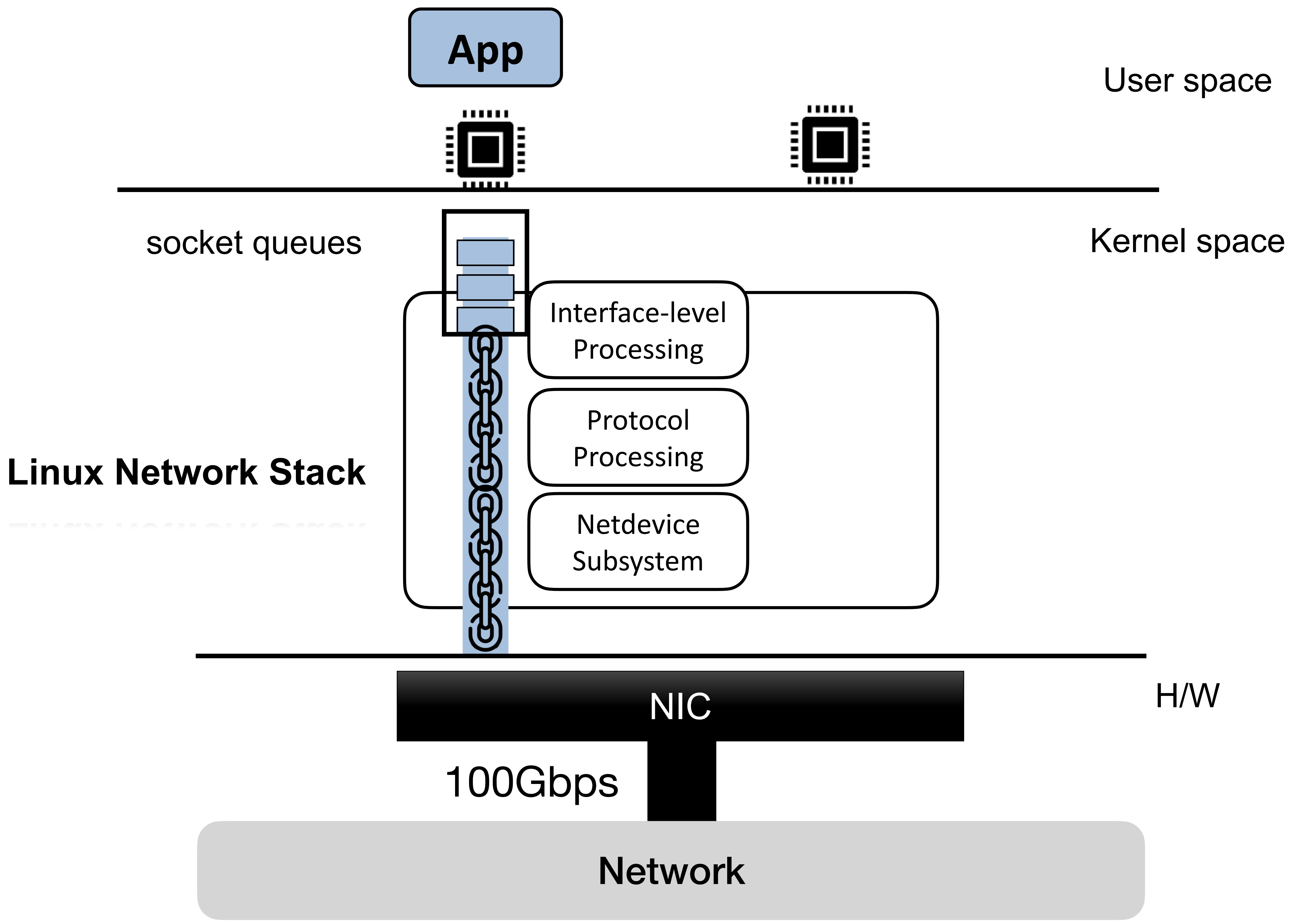
Resources for Interface-level Processing limited by number of application cores

Ideal: Dynamically scale Interface-level processing based on resource availability

Problem with Static and Dedicated Pipes

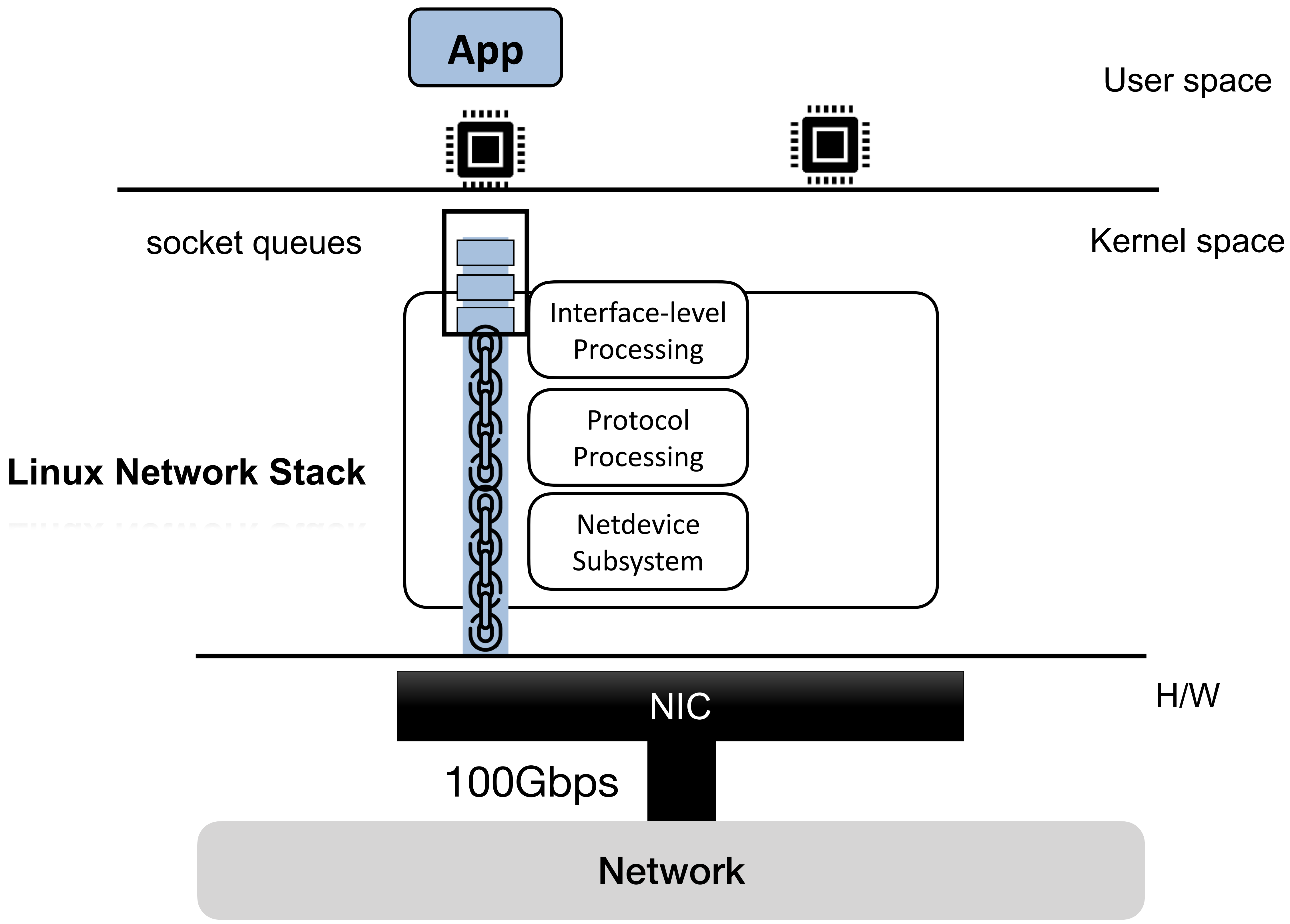


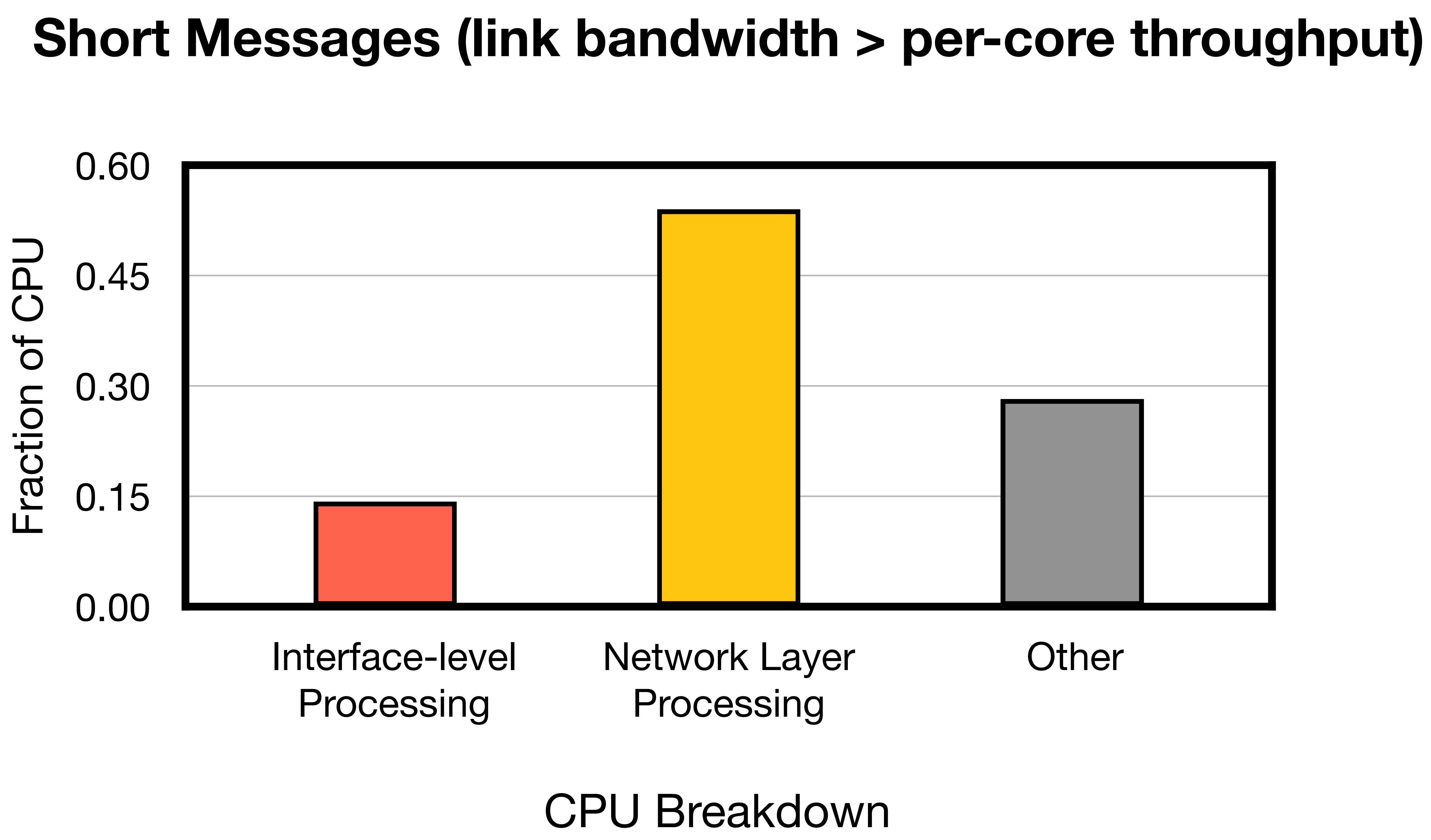
Problem with Static and Dedicated Pipes



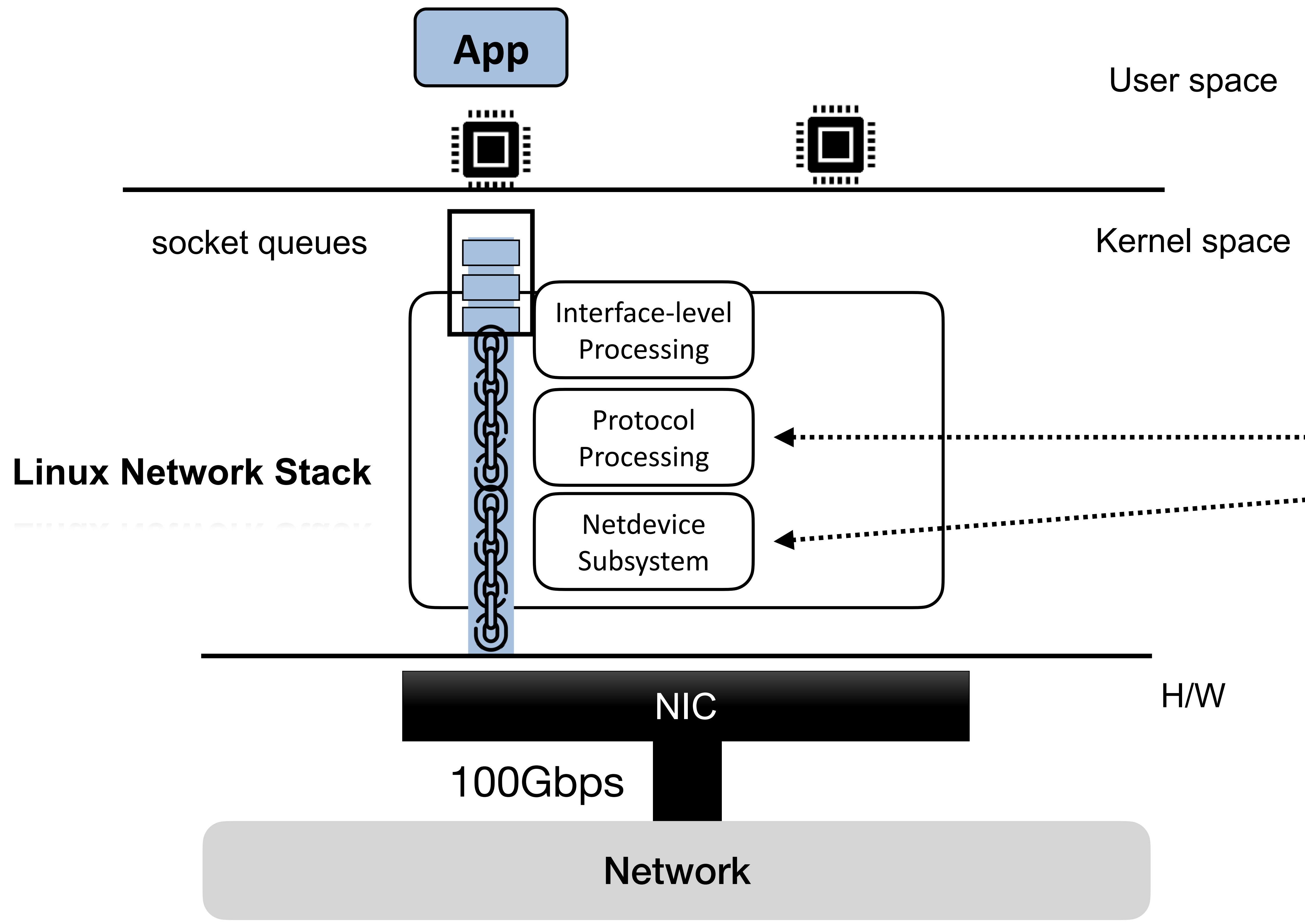
Short Messages (link bandwidth > per-core throughput)

Problem with Static and Dedicated Pipes

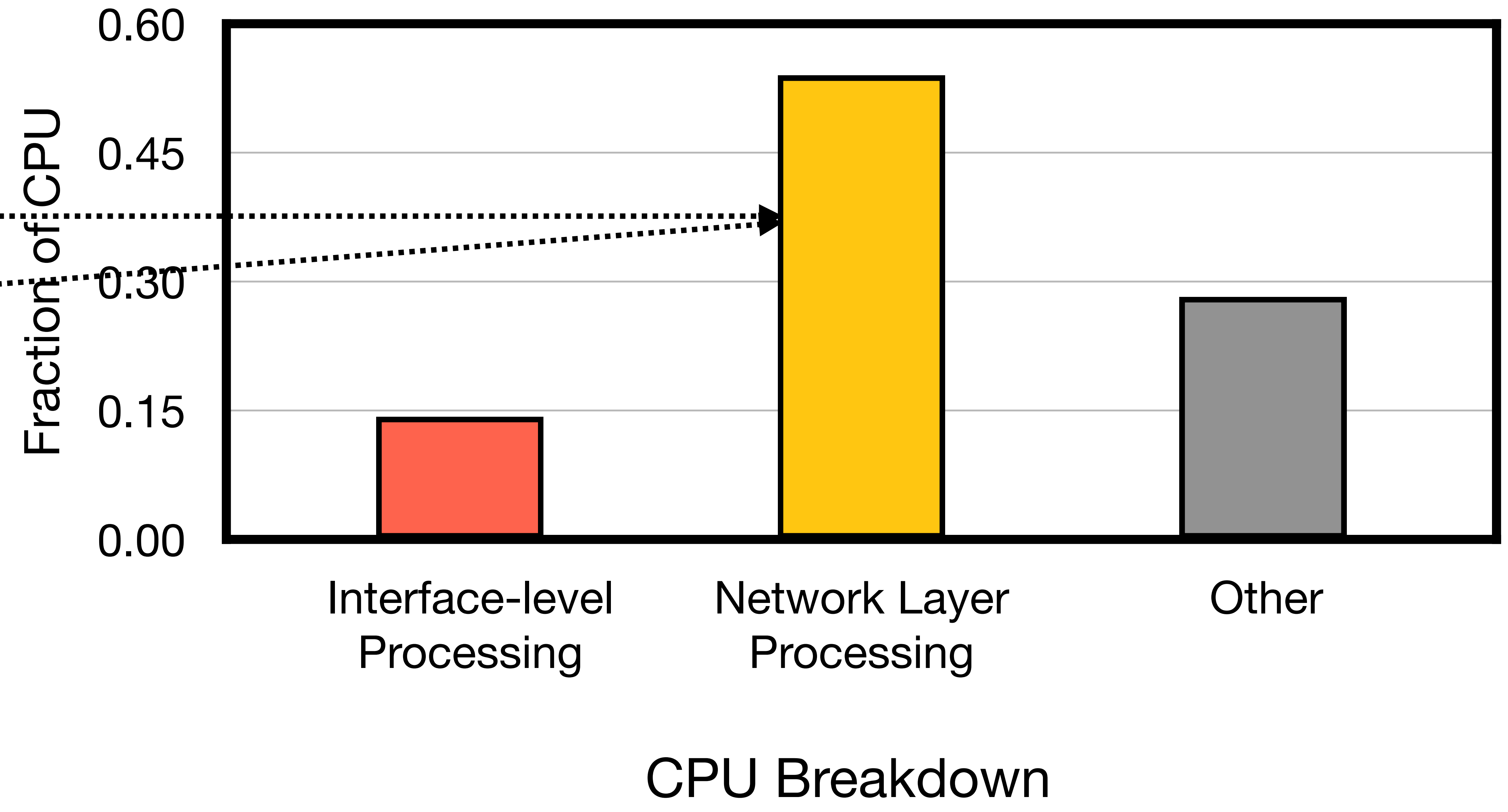




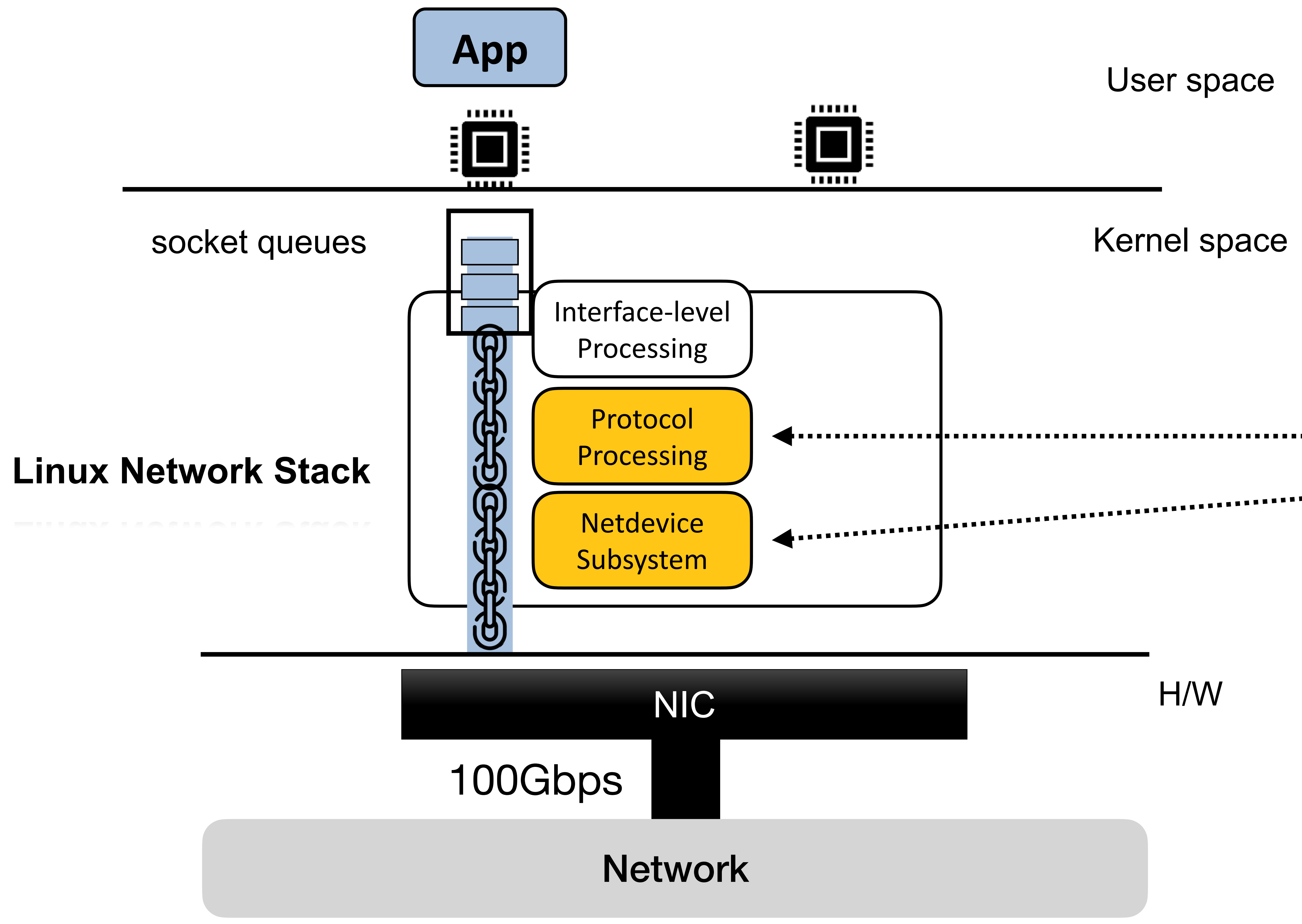
Problem with Static and Dedicated Pipes



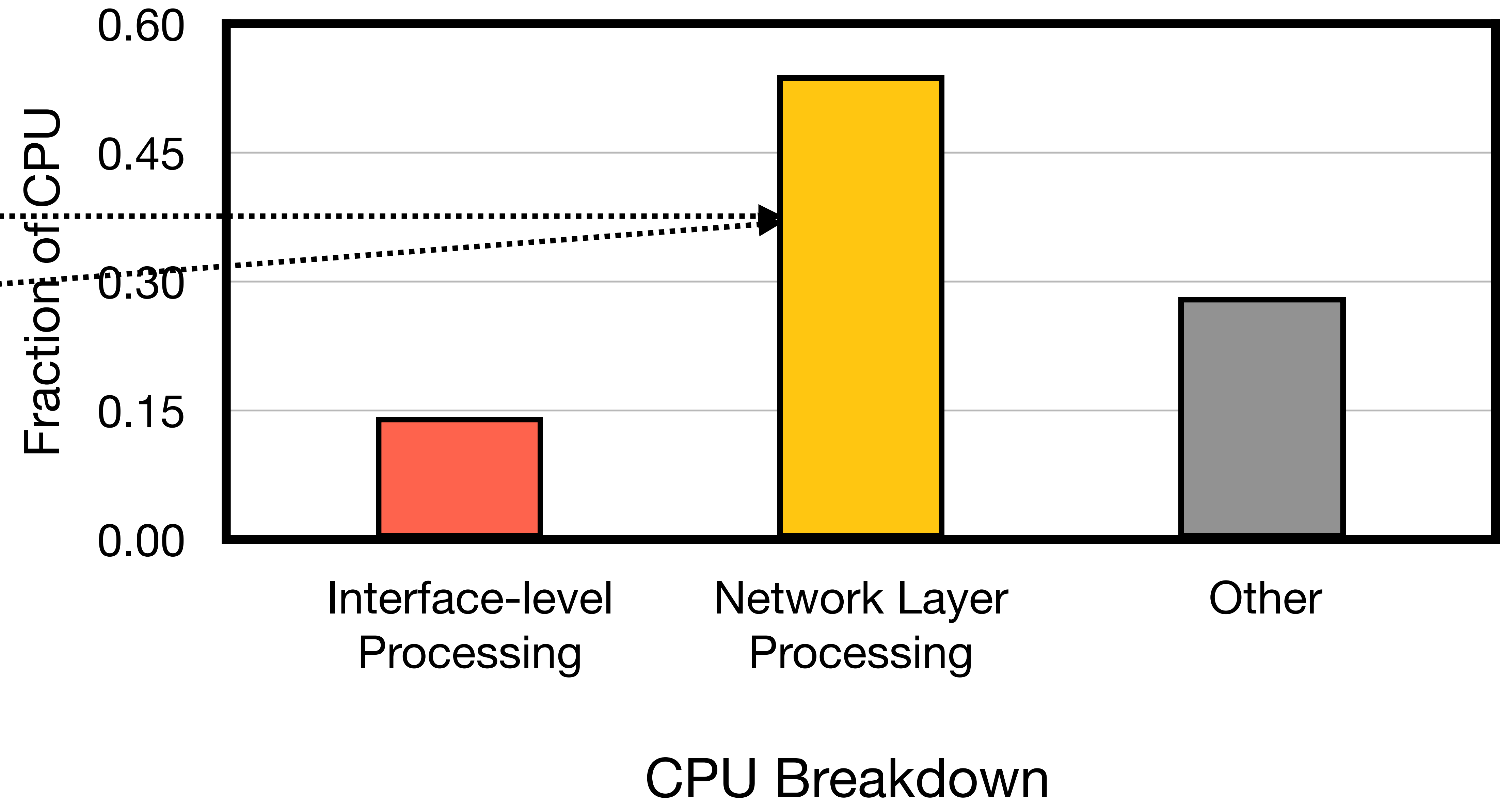
Short Messages (link bandwidth > per-core throughput)



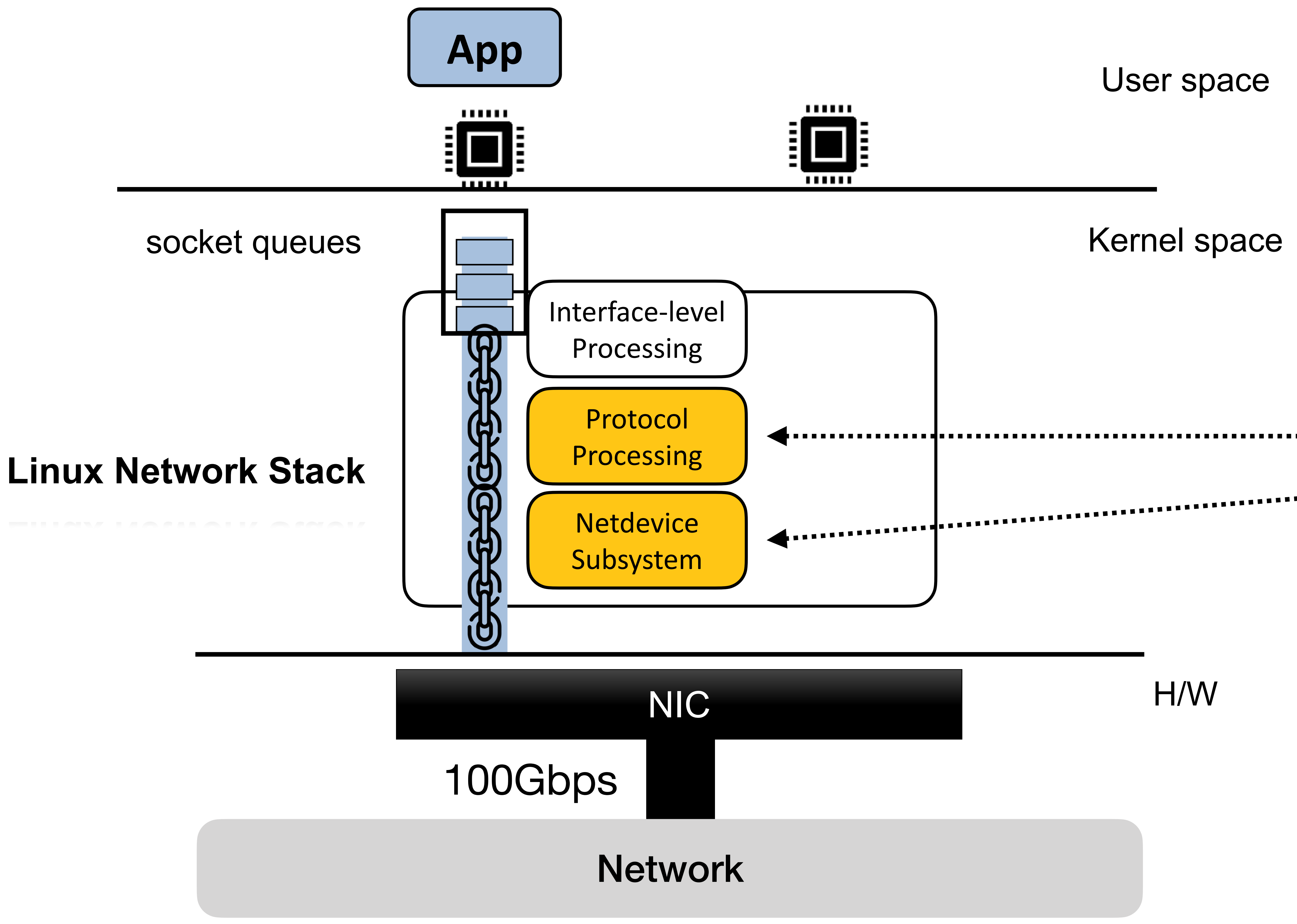
Problem with Static and Dedicated Pipes

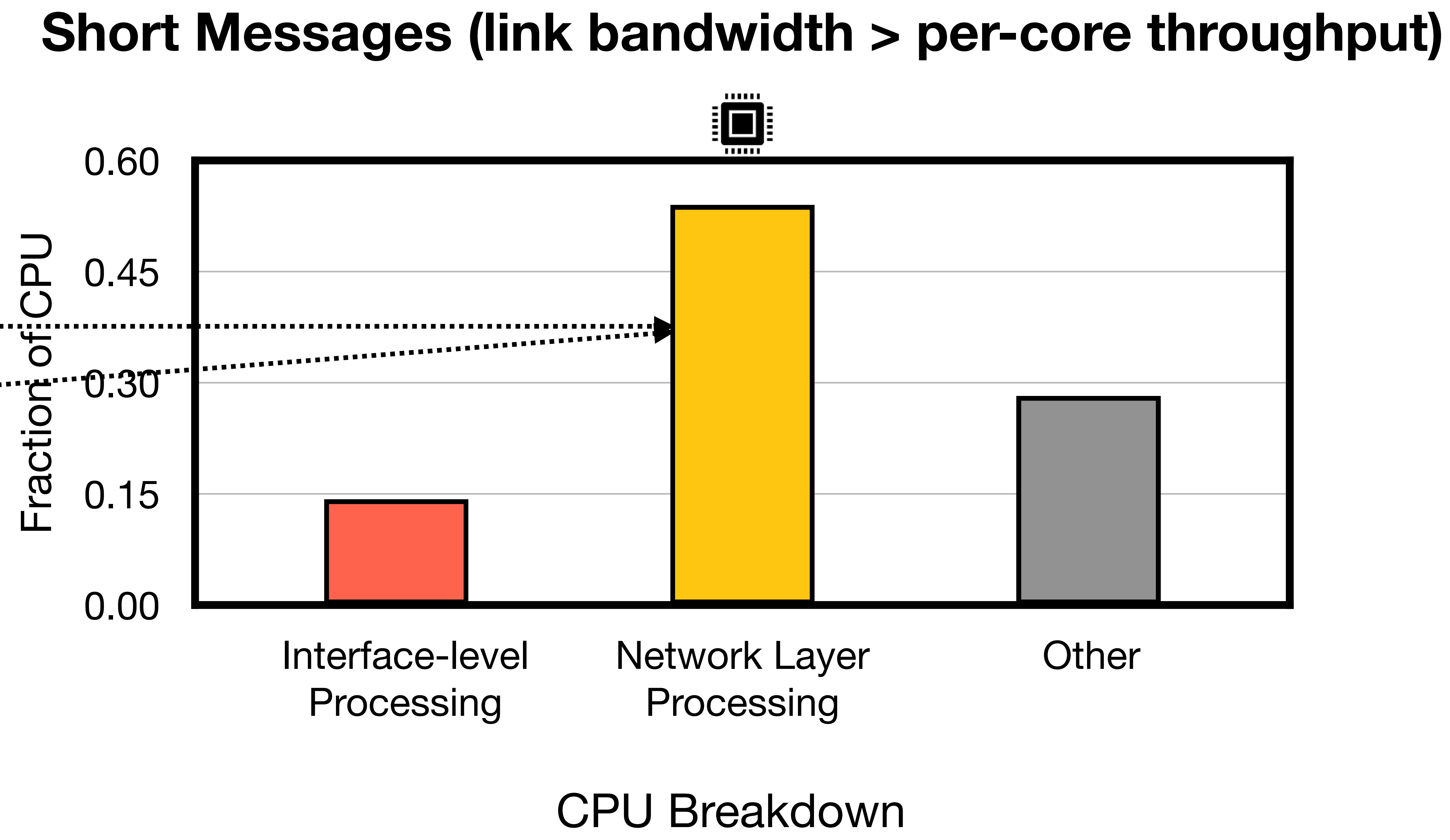


Short Messages (link bandwidth > per-core throughput)



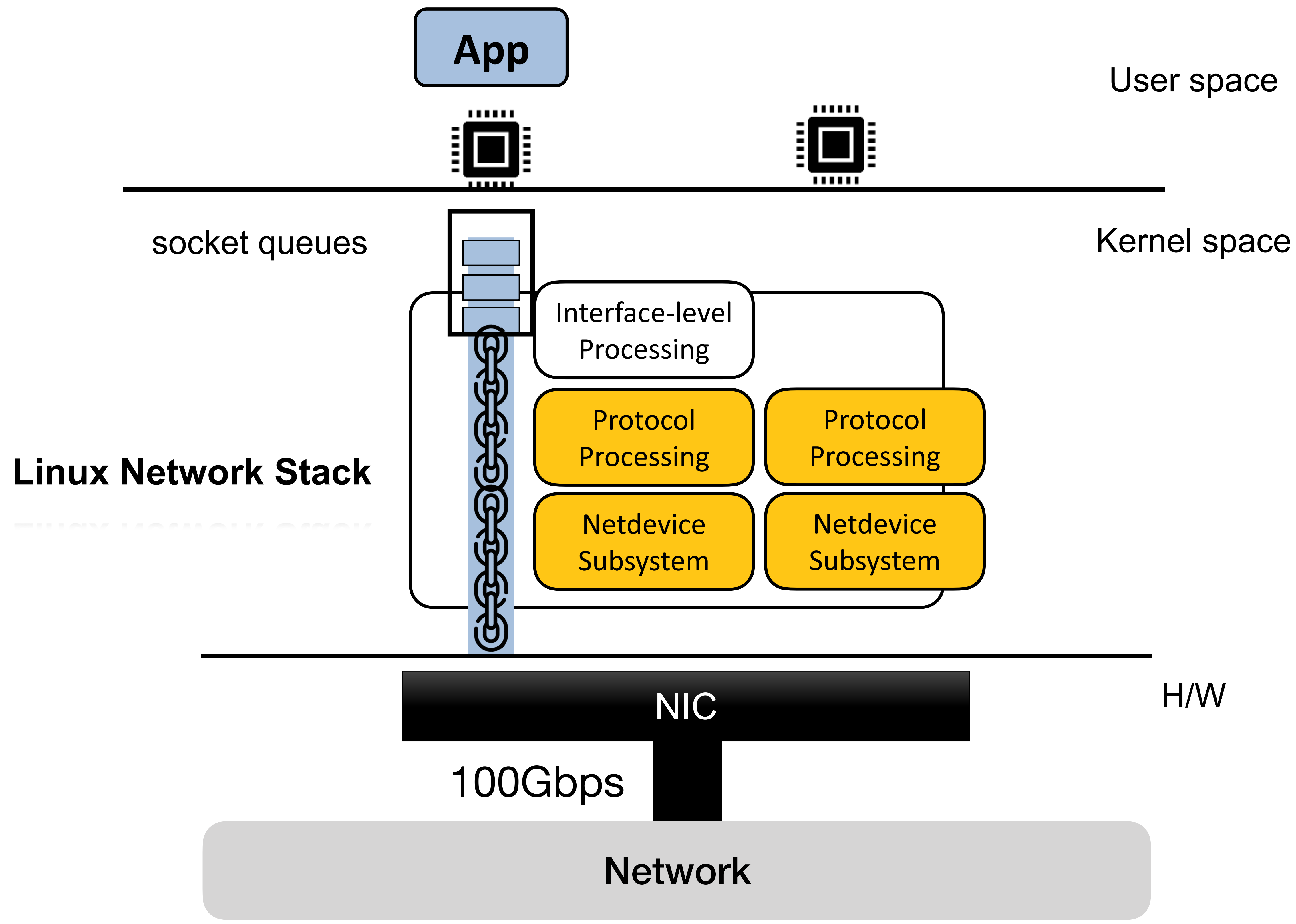
Problem with Static and Dedicated Pipes

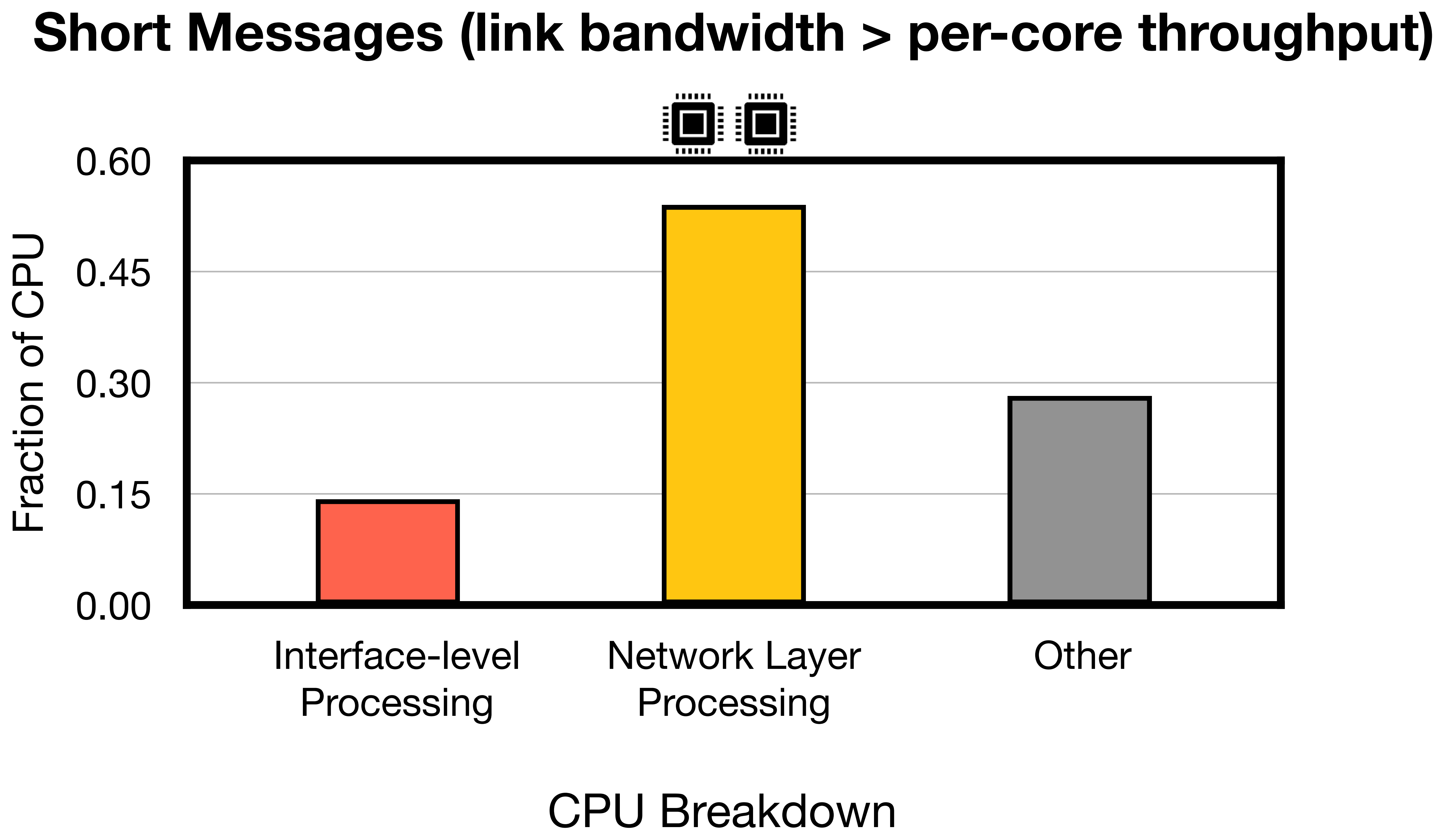




Resources for Network Layer Processing limited by number of connections

Problem with Static and Dedicated Pipes

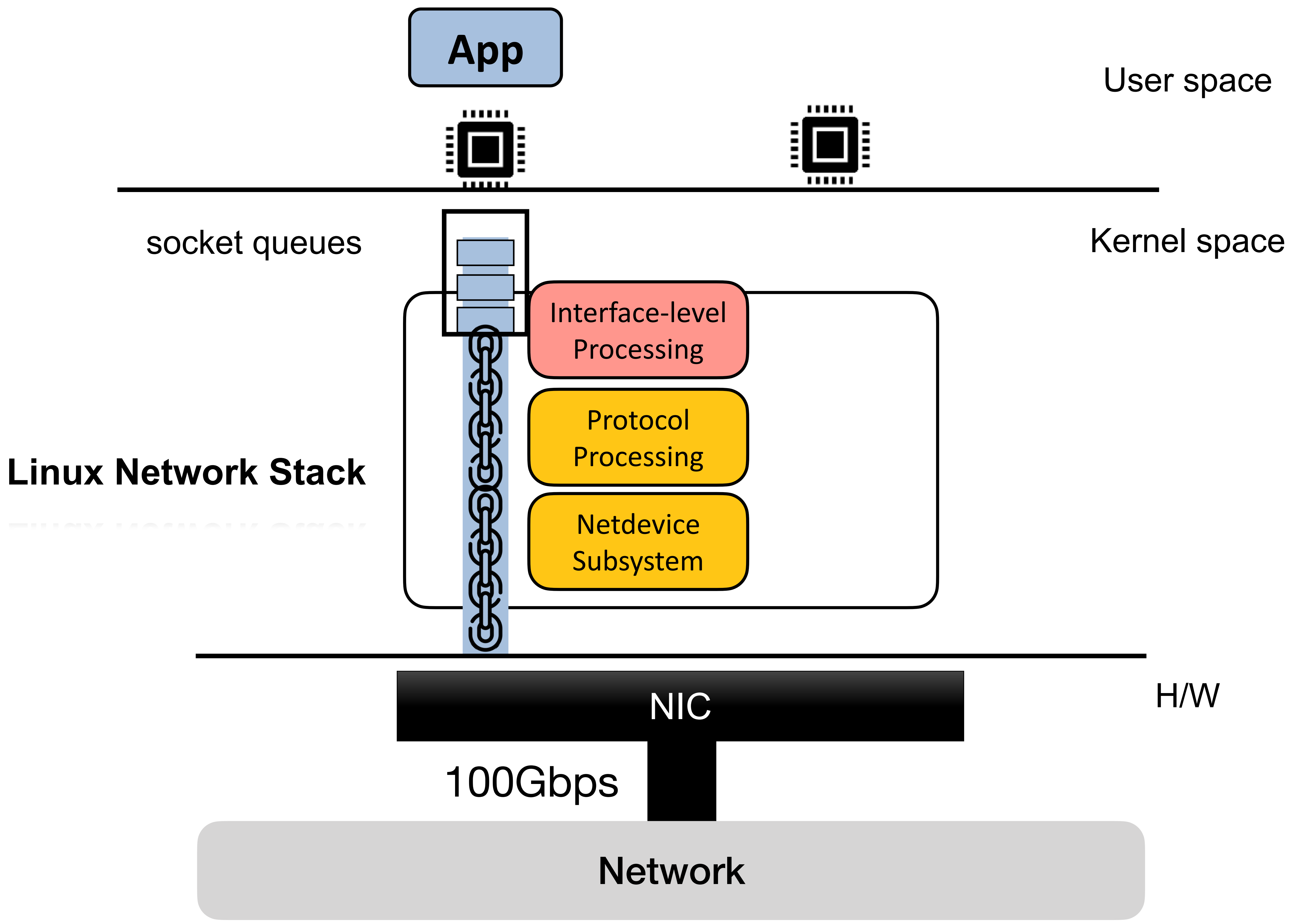




Resources for Network Layer Processing limited by number of connections

Ideal: Dynamically scale Network Layer Processing based on resource availability

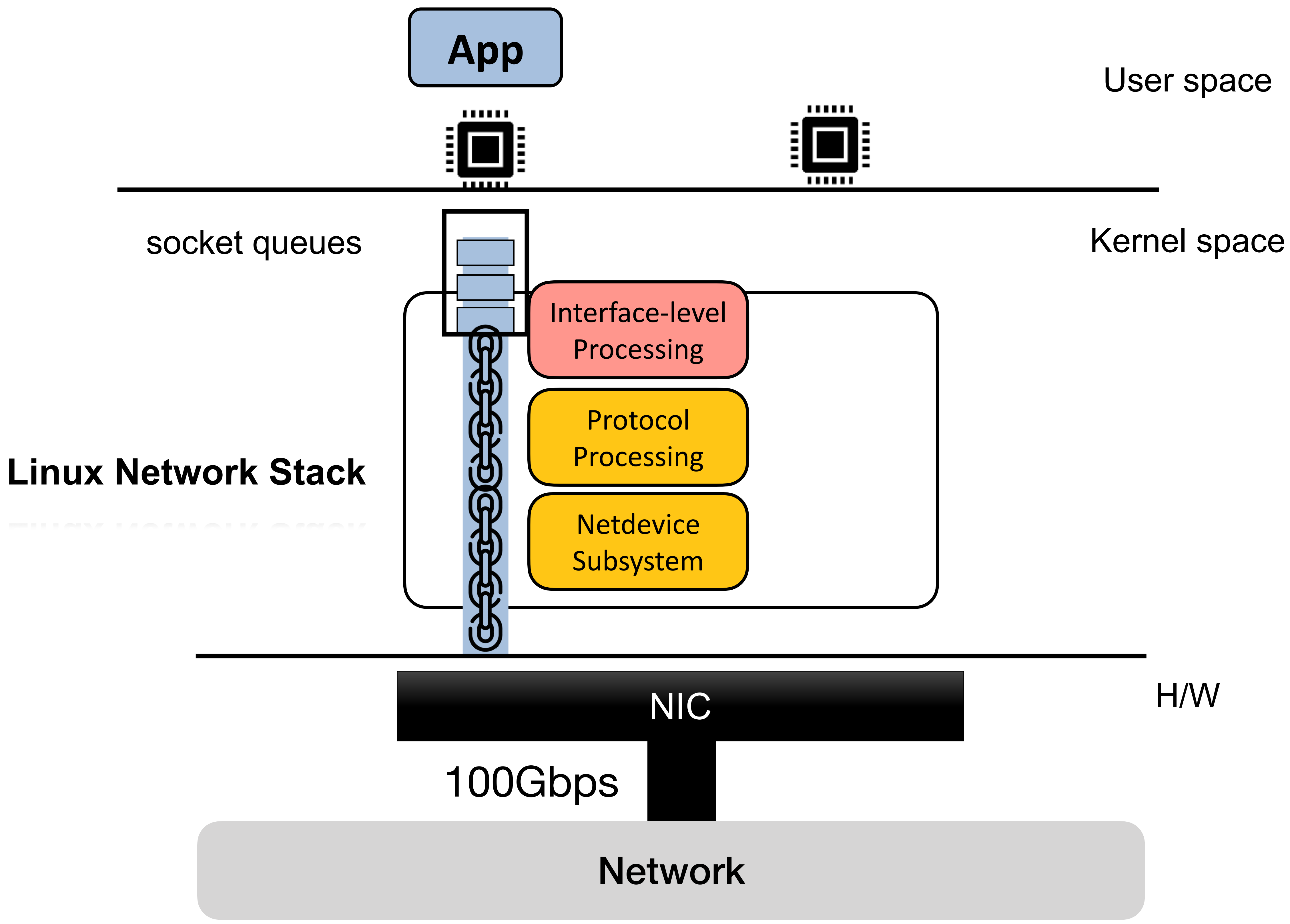
Problem with Static and Dedicated Pipes



Long Flows (link bandwidth > per-core throughput)

Short Messages (link bandwidth > per-core throughput)

Problem with Static and Dedicated Pipes

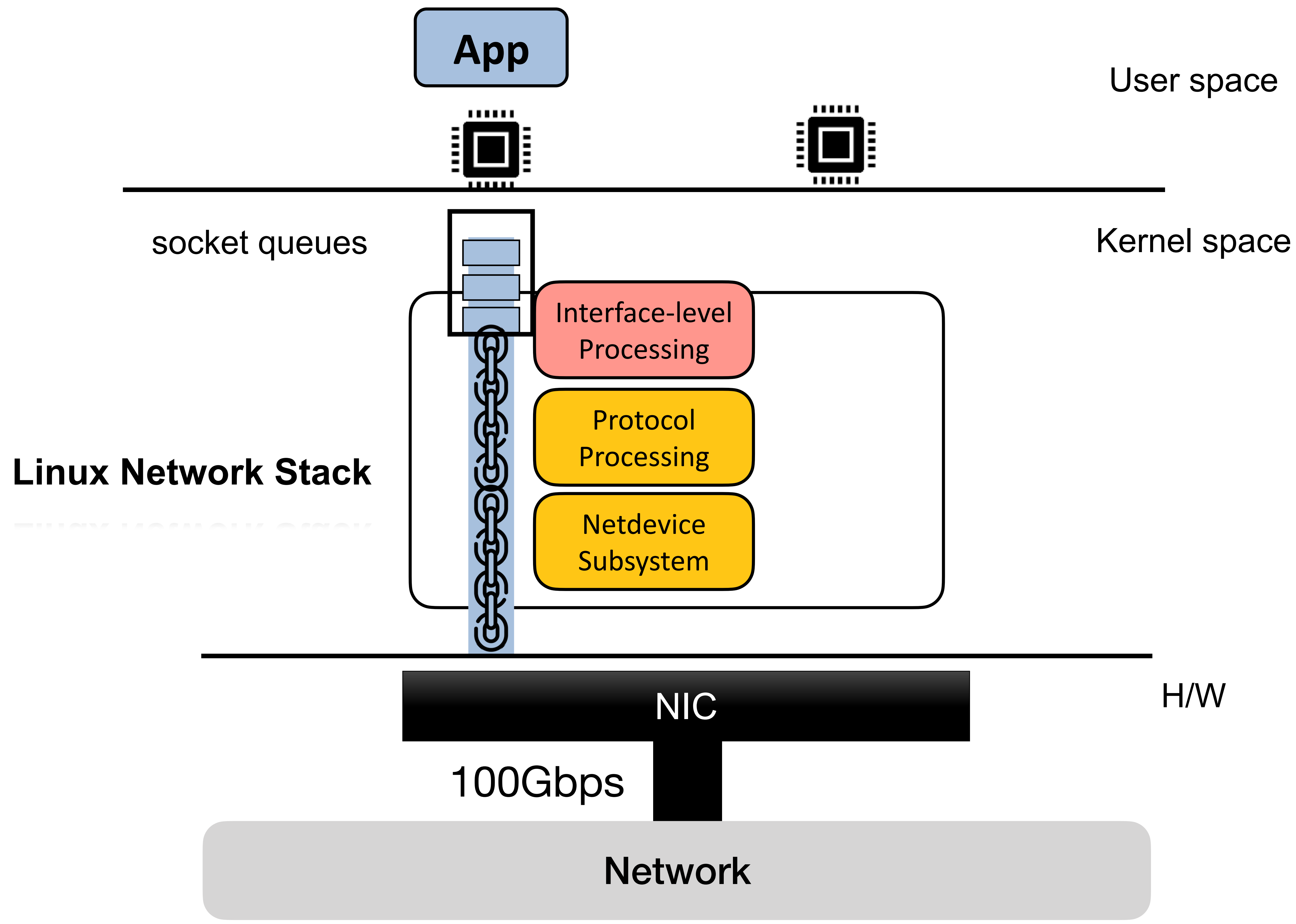


Long Flows (link bandwidth > per-core throughput)

Need more resources for Interface-level Processing

Short Messages (link bandwidth > per-core throughput)

Problem with Static and Dedicated Pipes



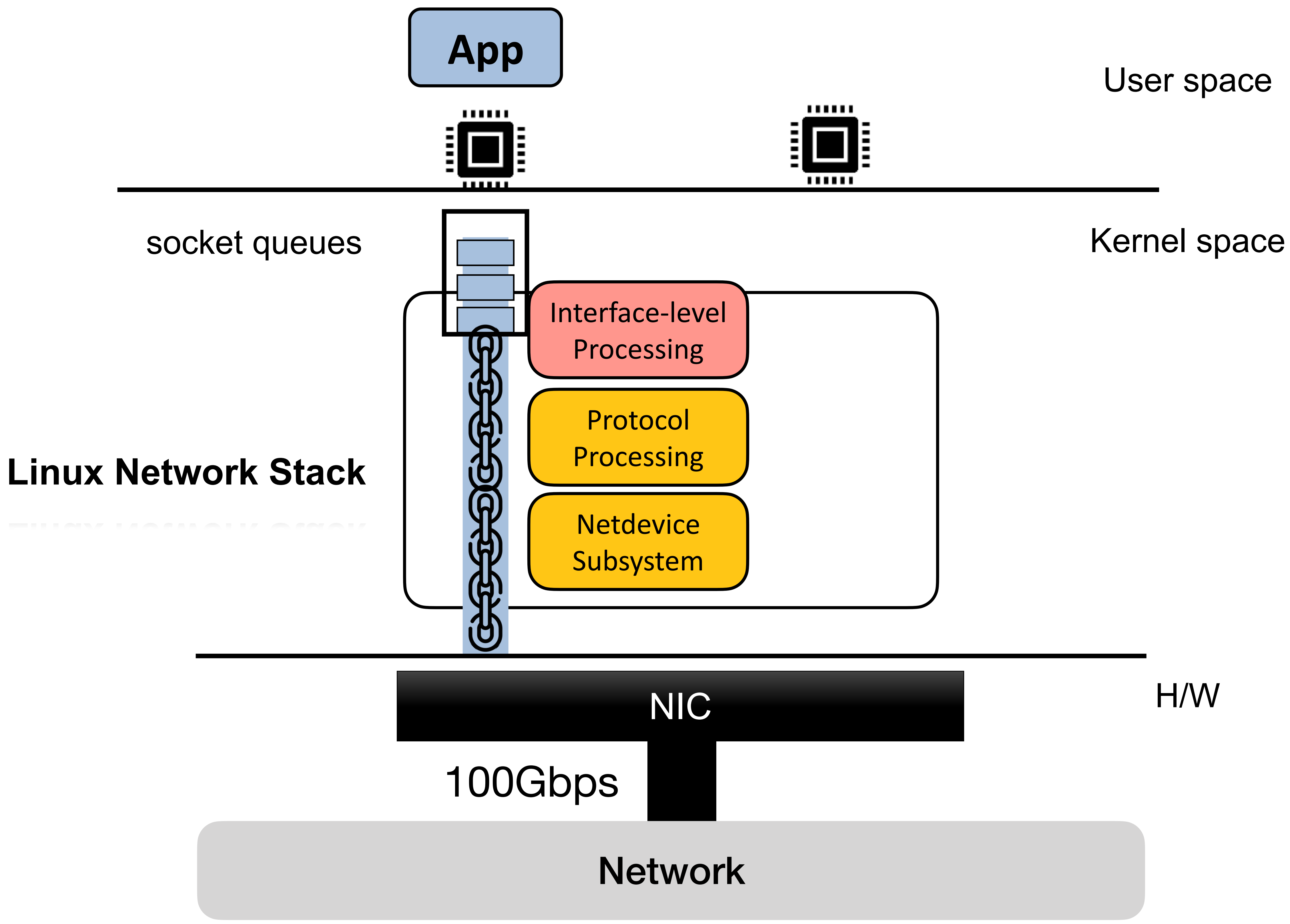
Long Flows (link bandwidth > per-core throughput)

Need more resources for Interface-level Processing

Short Messages (link bandwidth > per-core throughput)

Need more resources for Network Layer Processing

Problem with Static and Dedicated Pipes



Long Flows (link bandwidth > per-core throughput)

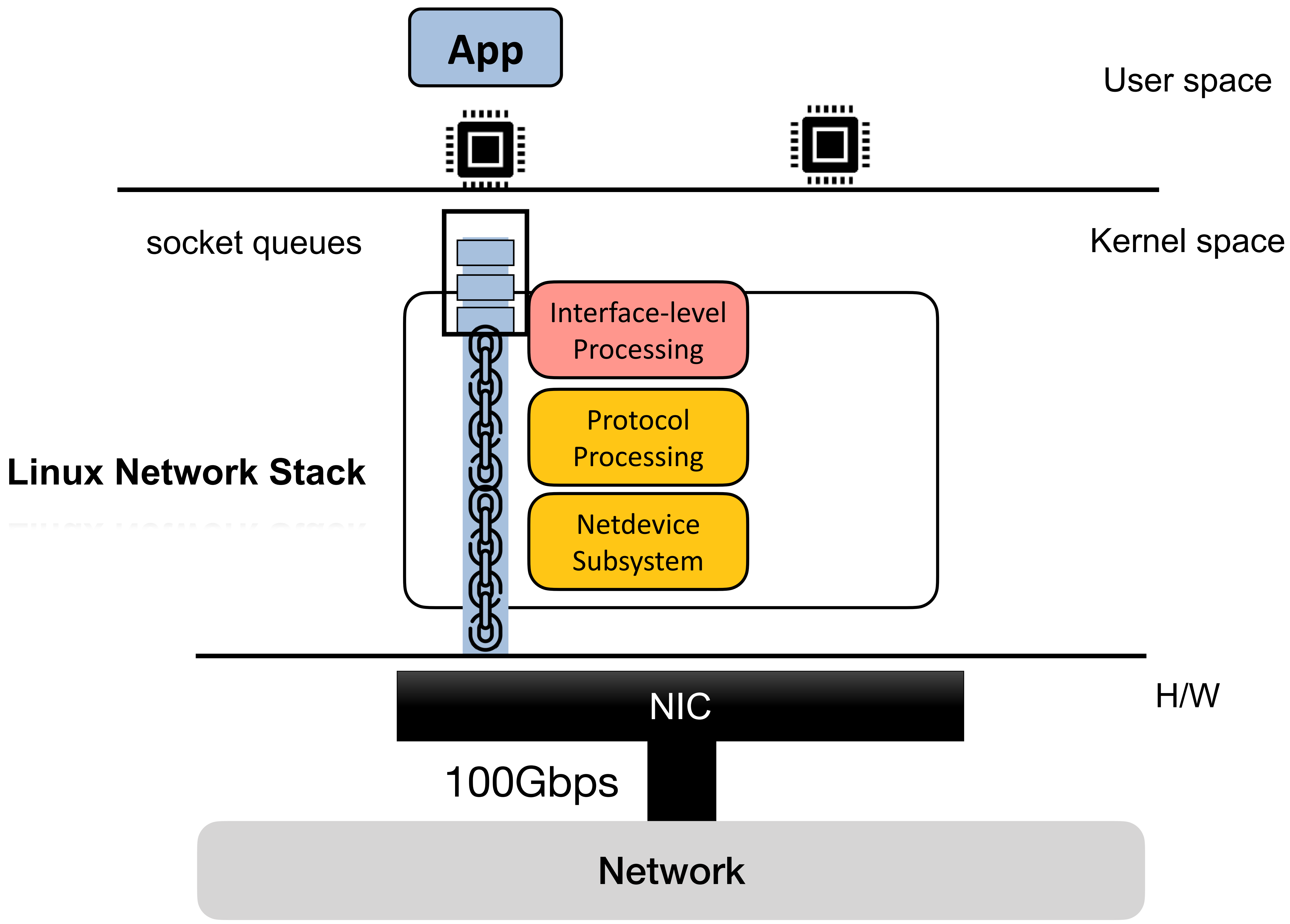
Need more resources for Interface-level Processing

Short Messages (link bandwidth > per-core throughput)

Need more resources for Network Layer Processing

Different applications can have bottlenecks at different parts of the pipeline

Problem with Static and Dedicated Pipes



Long Flows (link bandwidth > per-core throughput)

Need more resources for Interface-level Processing

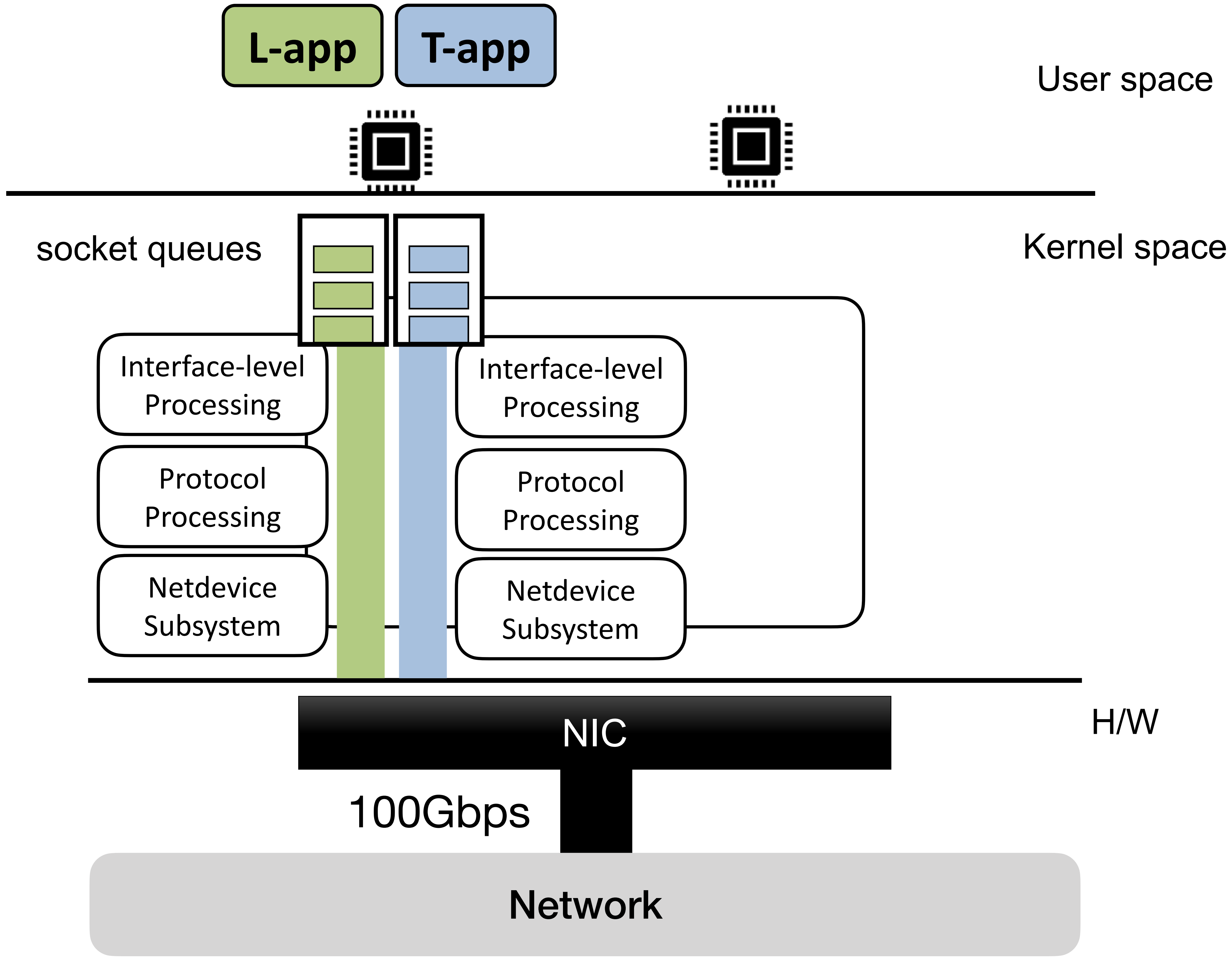
Short Messages (link bandwidth > per-core throughput)

Need more resources for Network Layer Processing

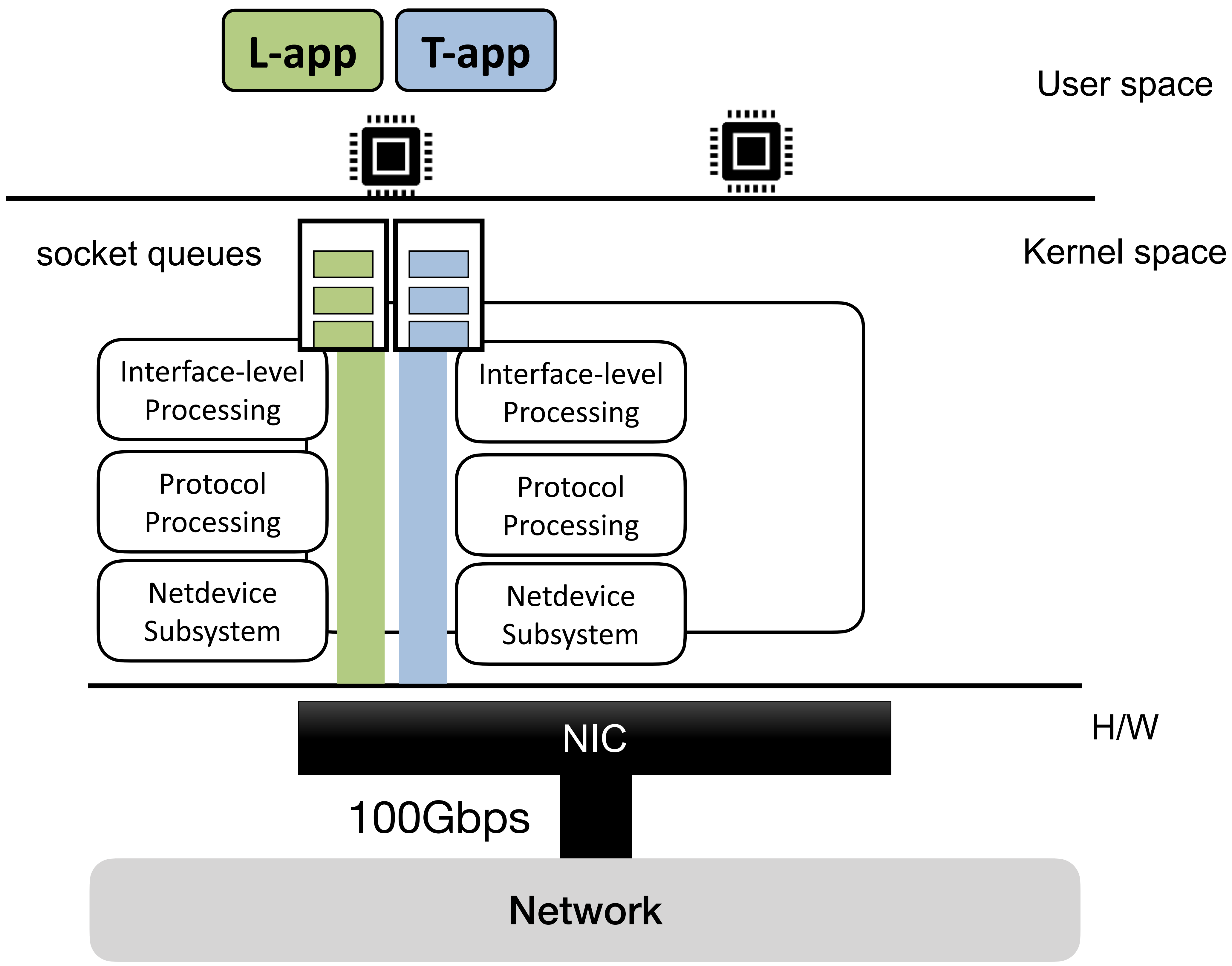
Different applications can have bottlenecks at different parts of the pipeline

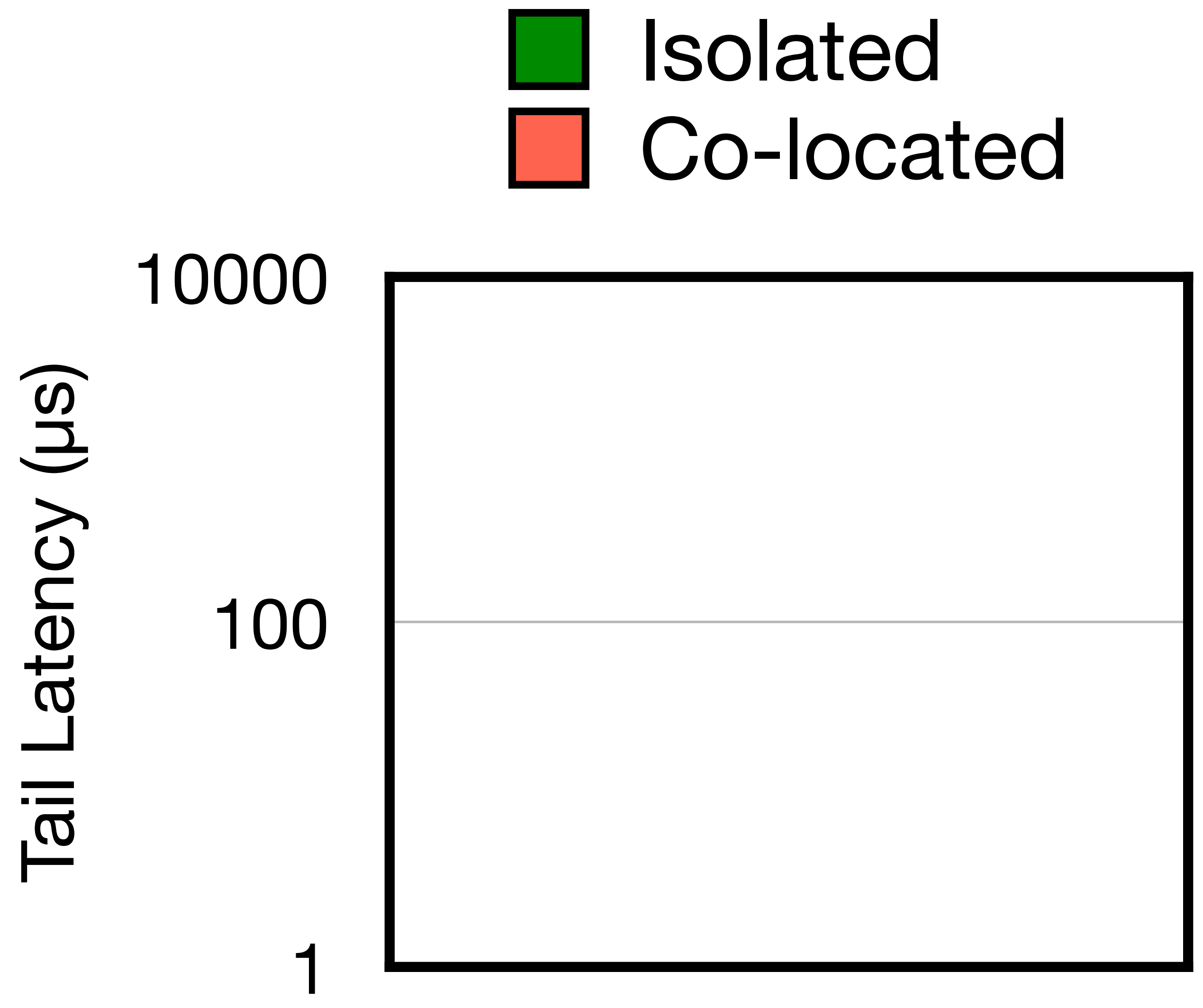
Ideal: Network stack should be able to independently allocate resources for different applications at different parts of the pipeline

Problem with Tightly Integrated Pipes

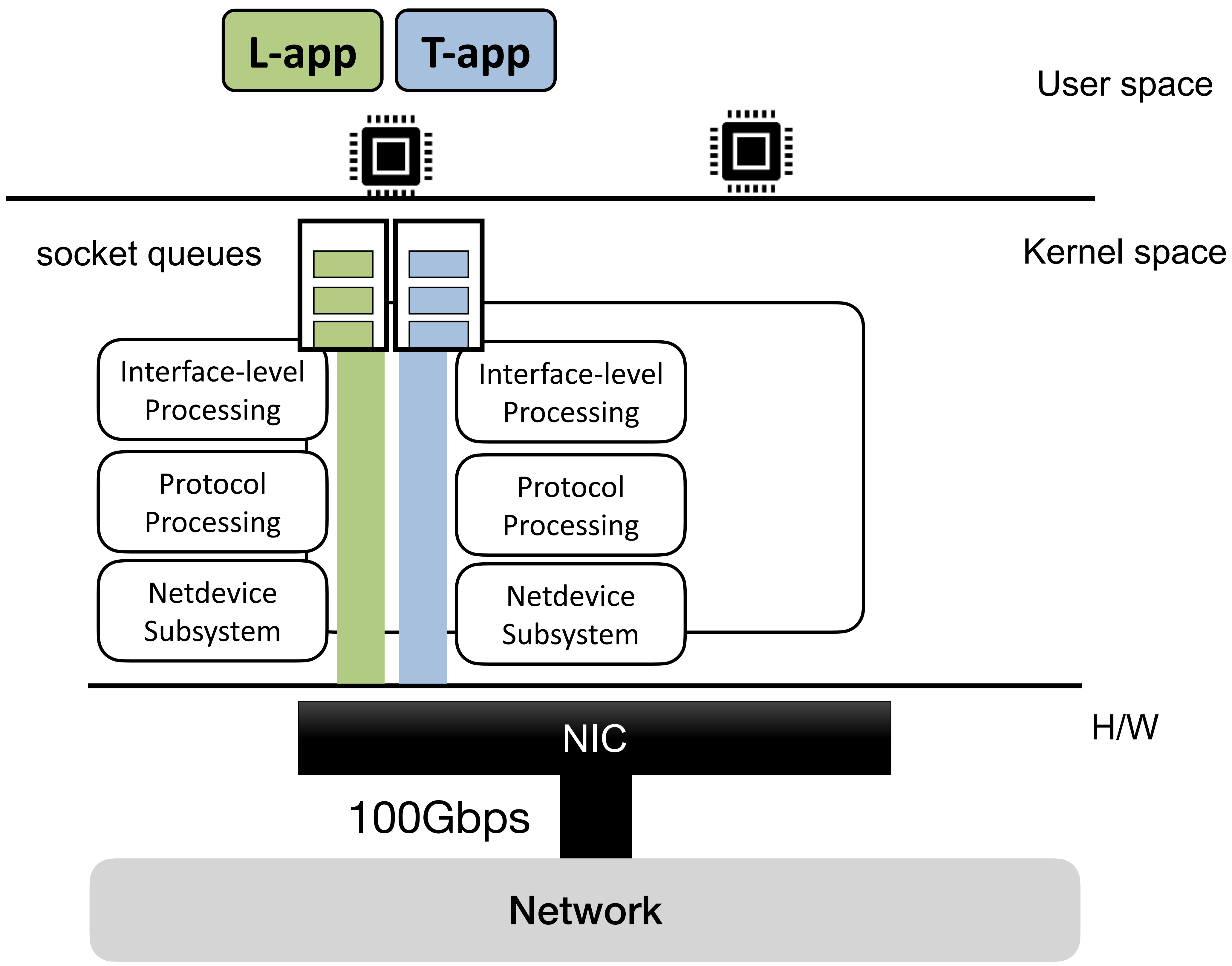


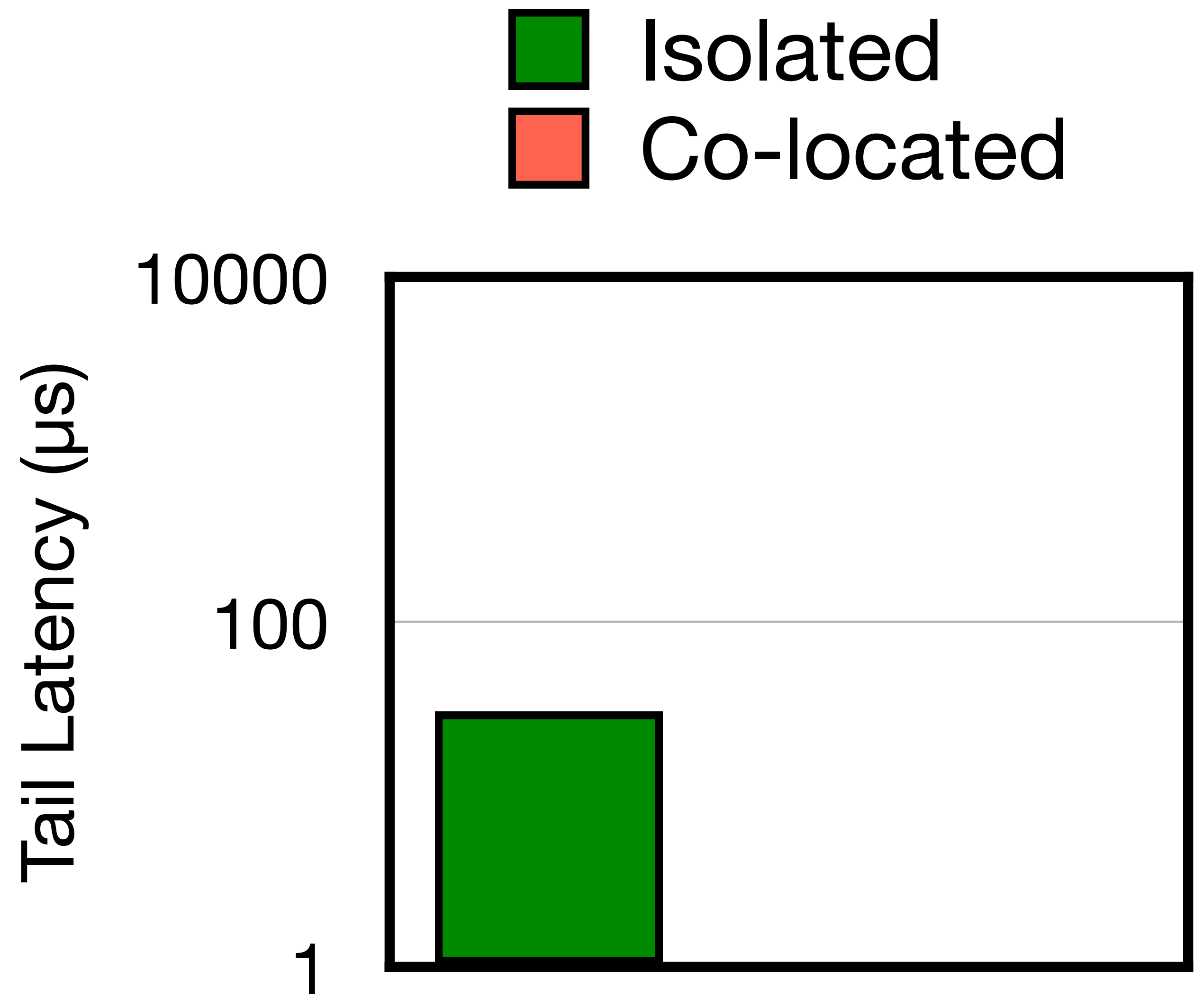
Problem with Tightly Integrated Pipes



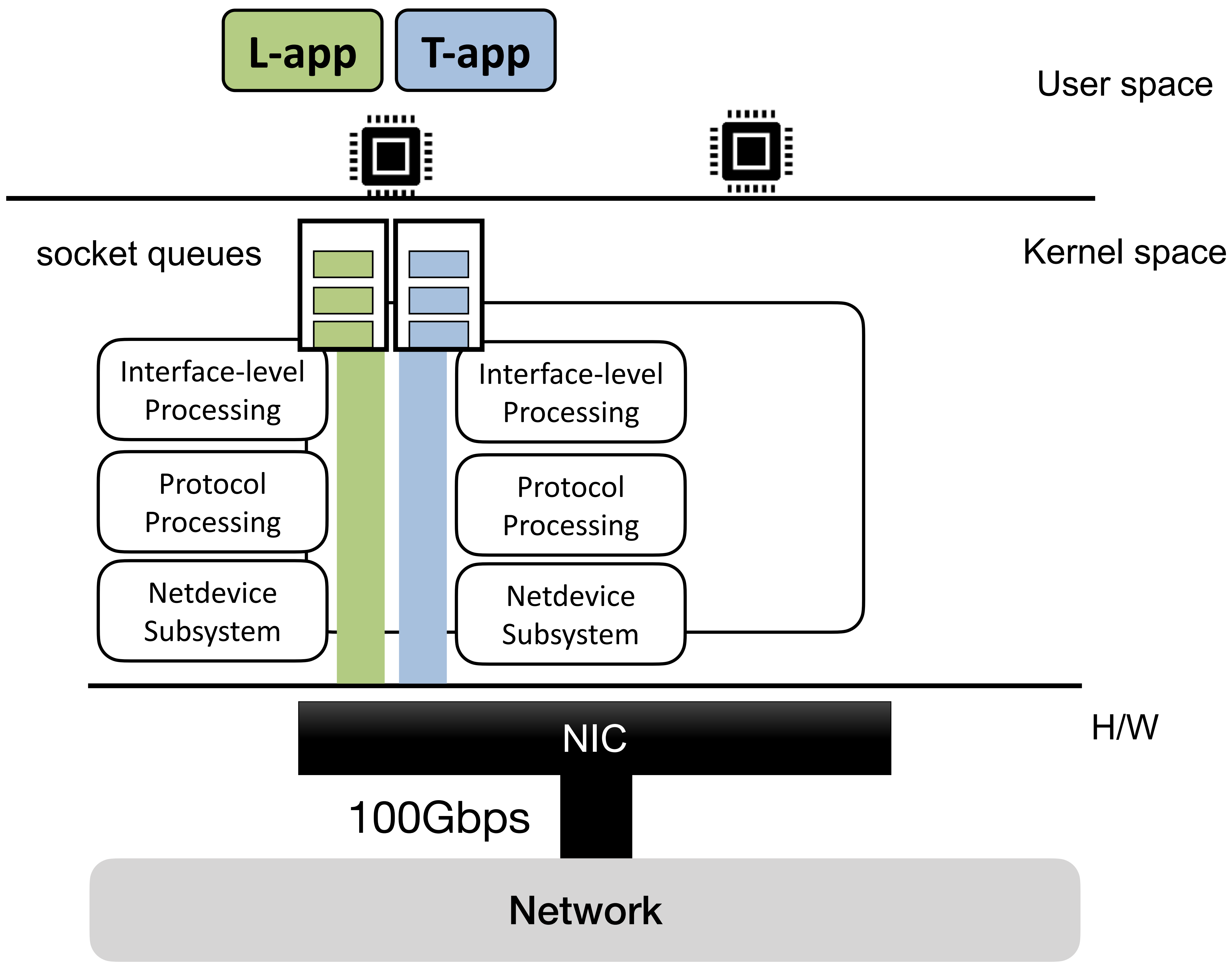


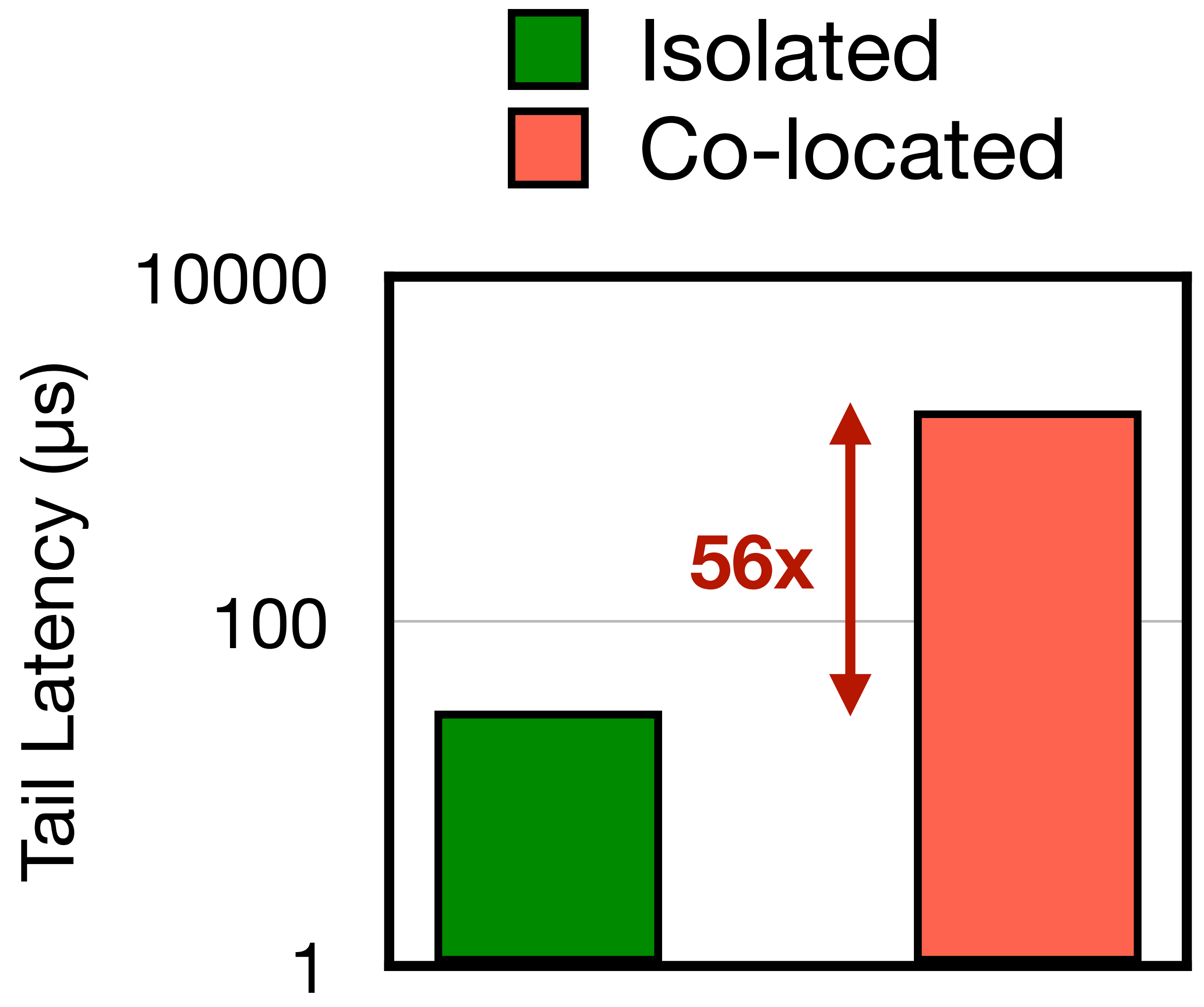
Problem with Tightly Integrated Pipes





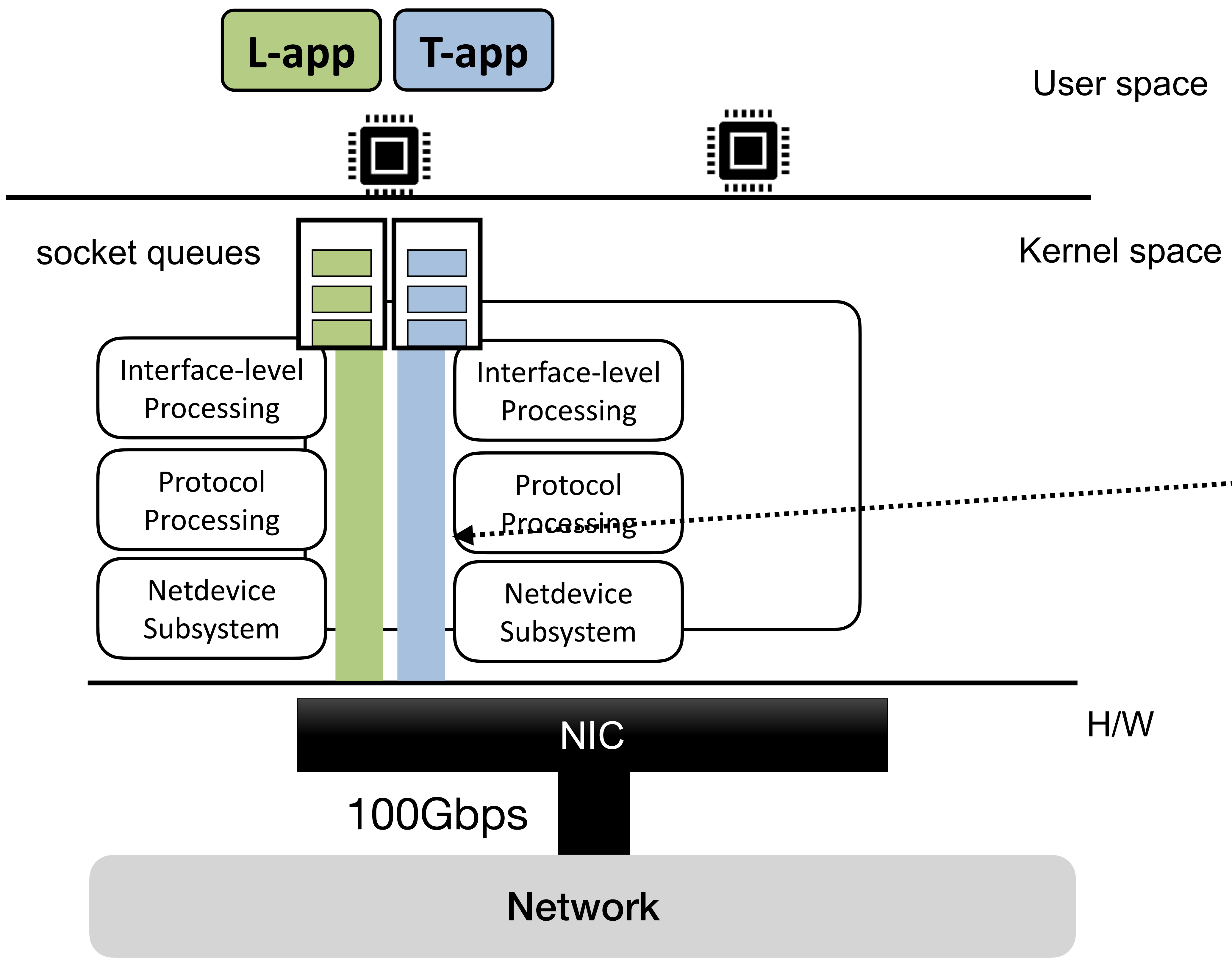
Problem with Tightly Integrated Pipes

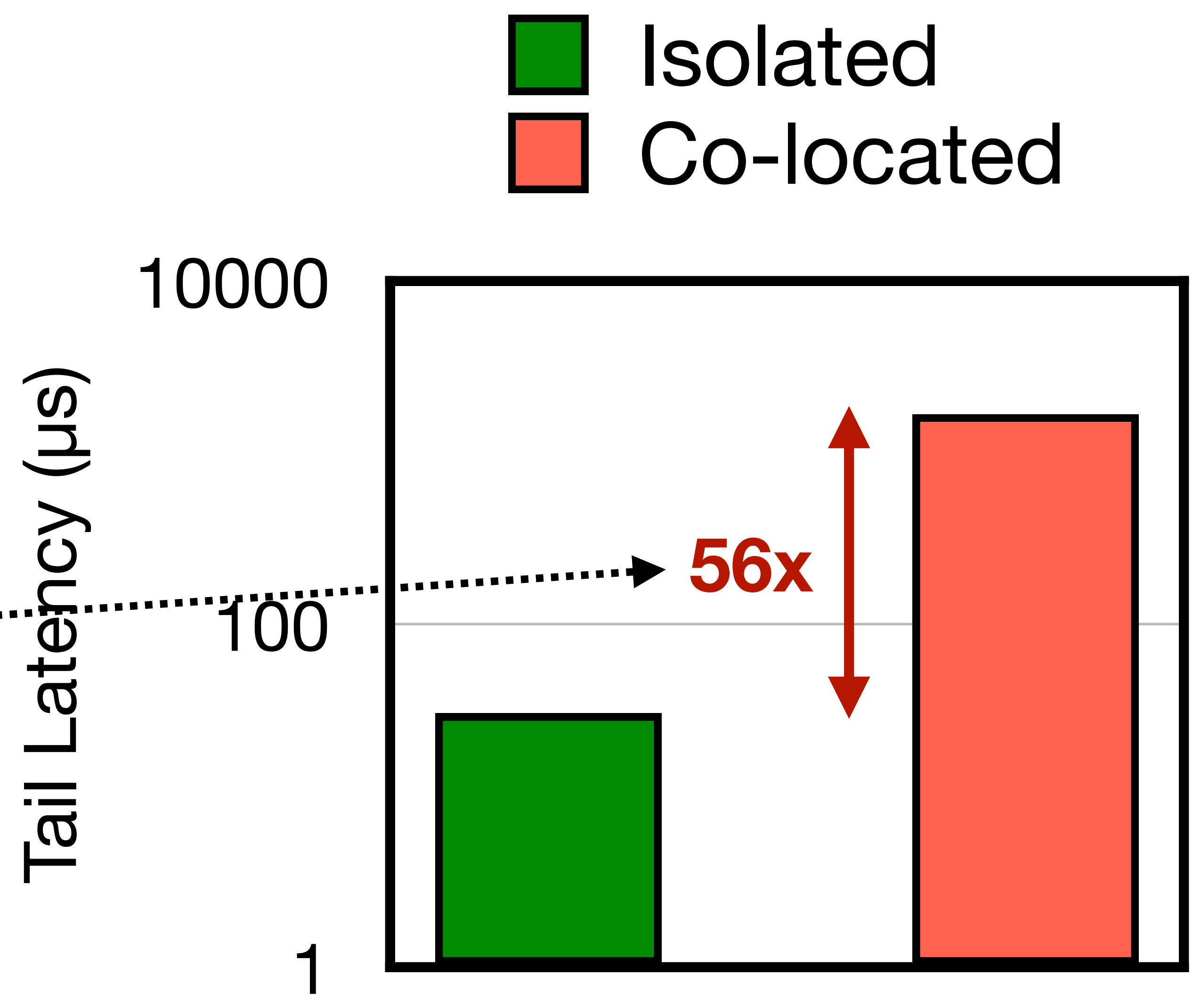




Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem

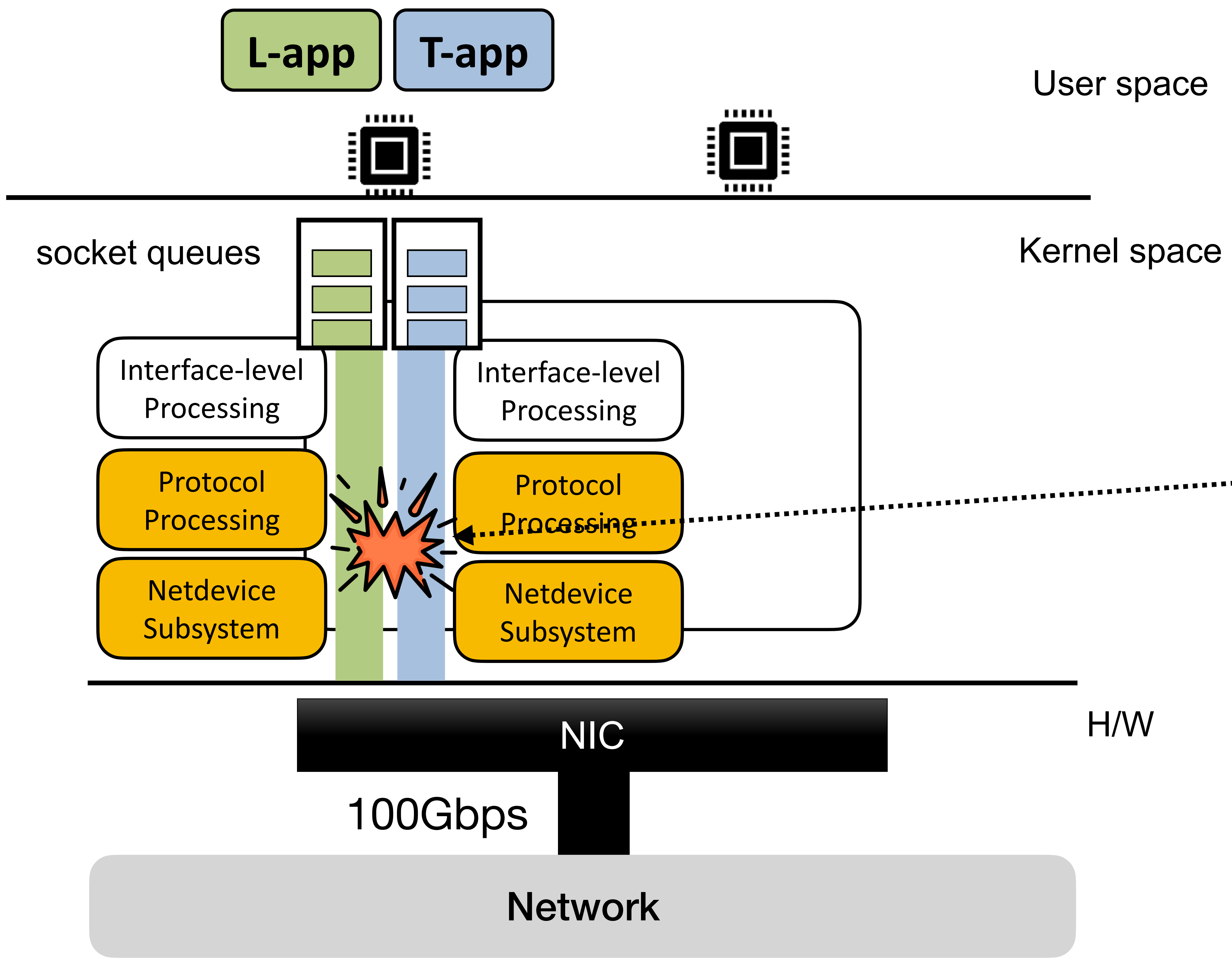
Problem with Tightly Integrated Pipes

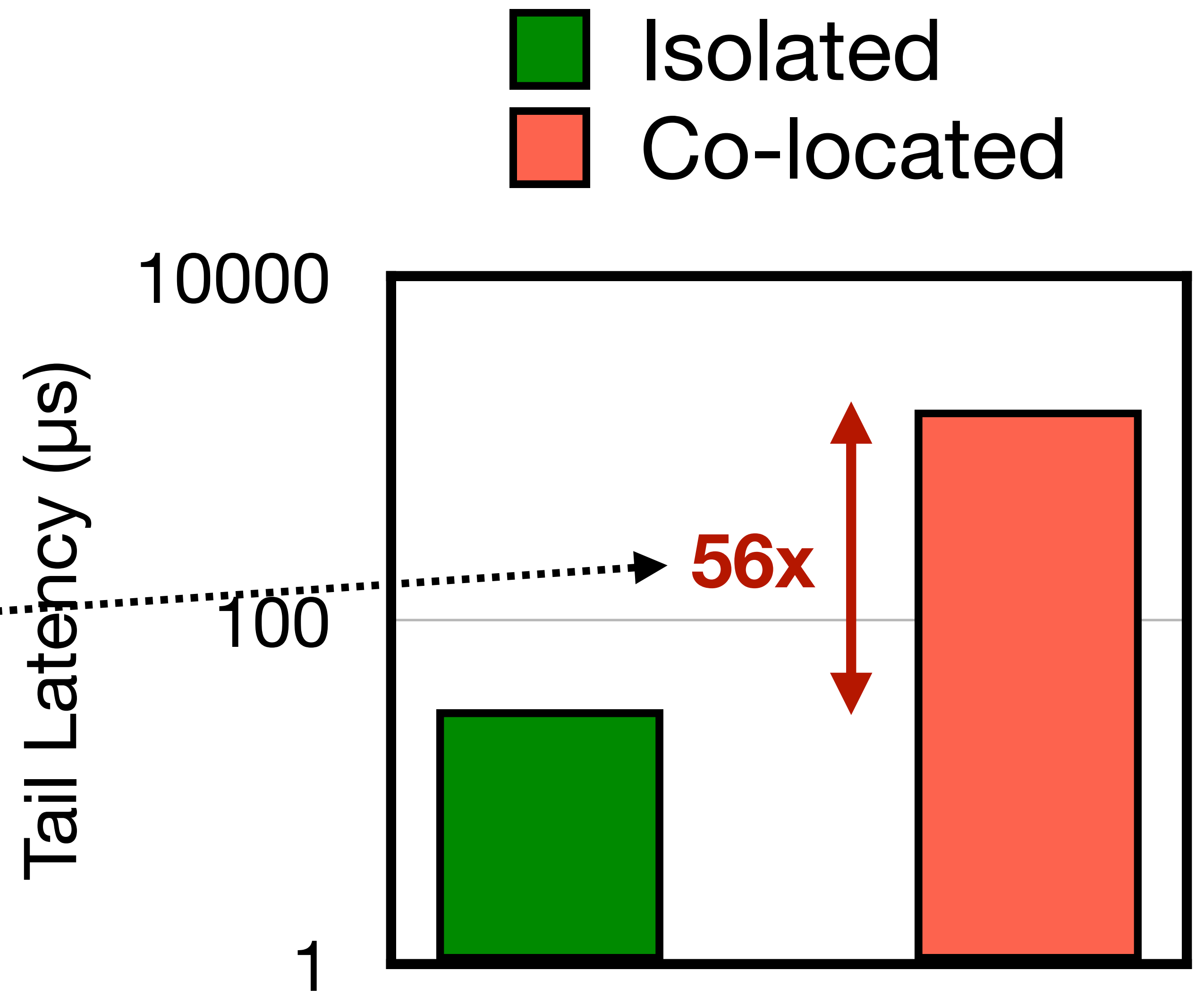




Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem

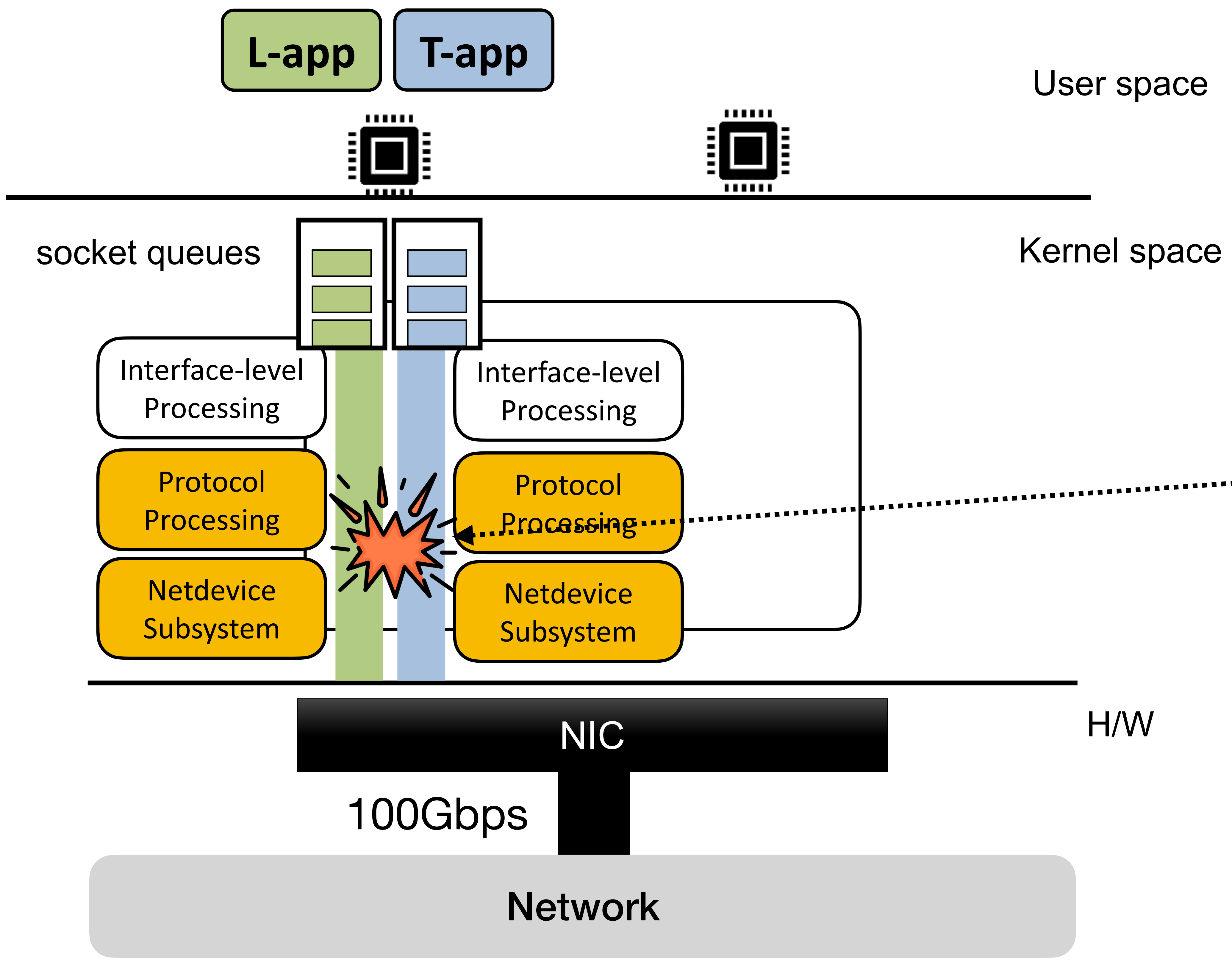
Problem with Tightly Integrated Pipes

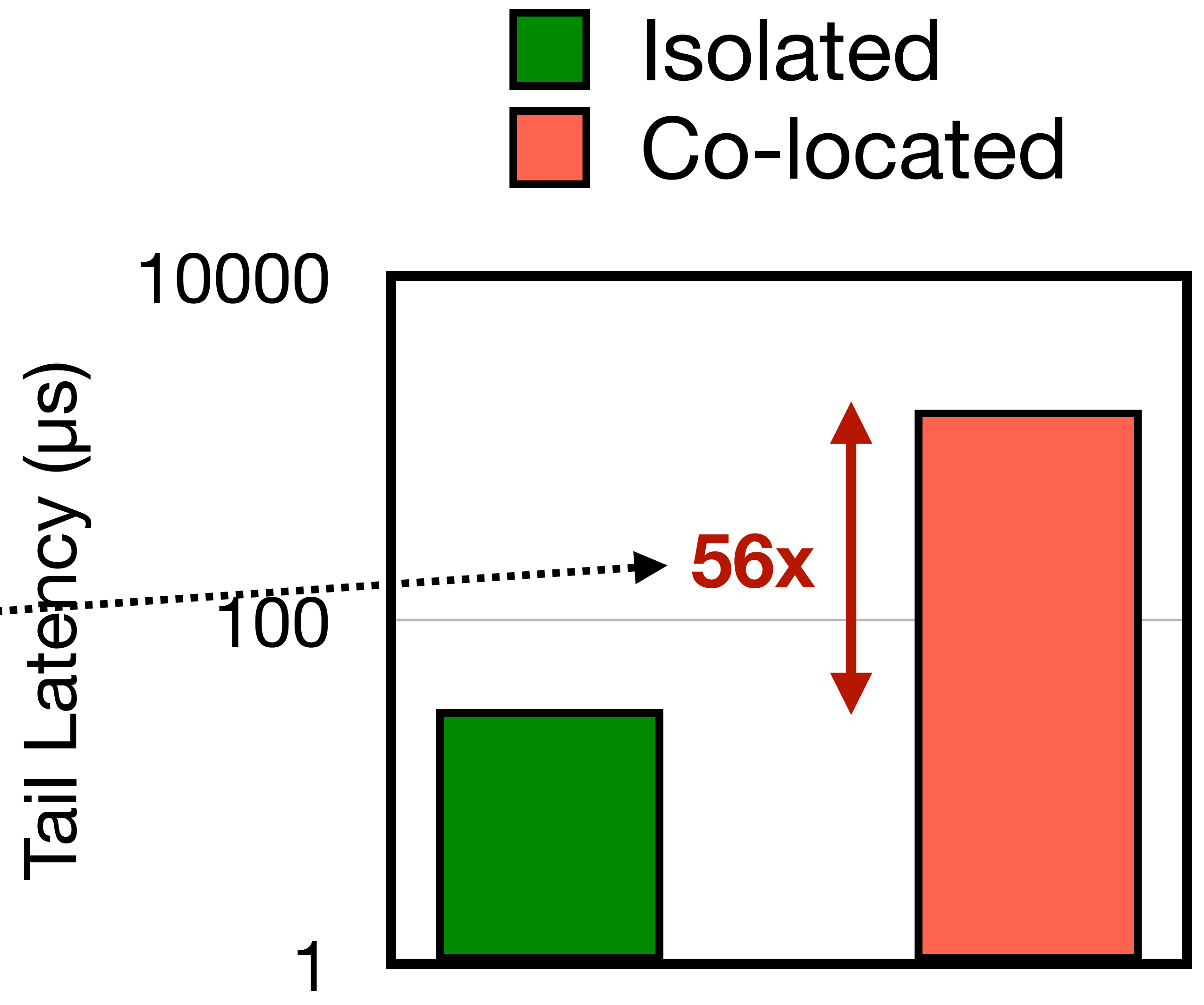




Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem

Problem with Tightly Integrated Pipes

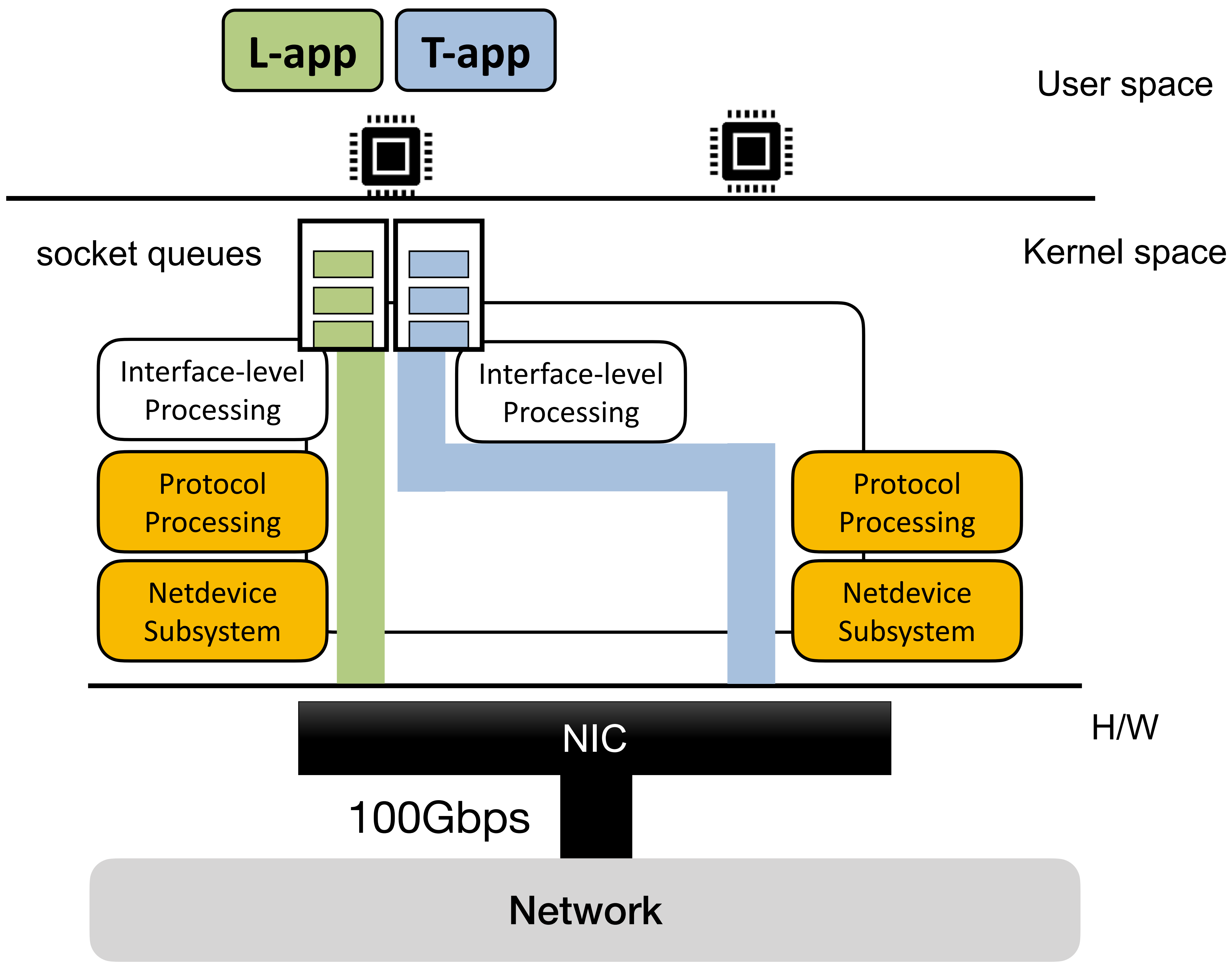




Prioritization mechanisms at NetDevice Subsystem & CPU scheduler do not solve the problem

Network Layer Processing coupled to core on which application runs

Problem with Tightly Integrated Pipes



Network Layer Processing coupled to core on which application runs

Ideal: Decouple Network Layer Processing from application cores

Rearchitecture is inevitable for Terabit Ethernet

Rearchitecture is inevitable for Terabit Ethernet

For ≥ 100Gbps networks, recent works have shown that bottlenecks have shifted to the Host

SIGCOMM'21	Understanding Host Network Stack Overheads
SIGCOMM'20	Swift: Delay is Simple and Effective for Congestion Control in the Datacenter
SIGCOMM'18	Understanding PCIe performance for end host networking

Rearchitecture is inevitable for Terabit Ethernet

For ≥ 100Gbps networks, recent works have shown that bottlenecks have shifted to the Host

SIGCOMM'21	Understanding Host Network Stack Overheads
SIGCOMM'20	Swift: Delay is Simple and Effective for Congestion Control in the Datacenter
SIGCOMM'18	Understanding PCIe performance for end host networking

	Internet / Early generation Datacenters
Dedicated, Tightly Integrated Static Pipelines	✓

Rearchitecture is inevitable for Terabit Ethernet

For ≥ 100Gbps networks, recent works have shown that bottlenecks have shifted to the Host

SIGCOMM'21	Understanding Host Network Stack Overheads
SIGCOMM'20	Swift: Delay is Simple and Effective for Congestion Control in the Datacenter
SIGCOMM'18	Understanding PCIe performance for end host networking

	Internet / Early generation Datacenters	Today's Datacenters us-latency, 100s of Gbps
Dedicated, Tightly Integrated Static Pipelines	✓	✗

Today's network stacks are on the brink of breakdown

Rearchitecture is inevitable for Terabit Ethernet

For ≥ 100Gbps networks, recent works have shown that bottlenecks have shifted to the Host

SIGCOMM'21	Understanding Host Network Stack Overheads
SIGCOMM'20	Swift: Delay is Simple and Effective for Congestion Control in the Datacenter
SIGCOMM'18	Understanding PCIe performance for end host networking

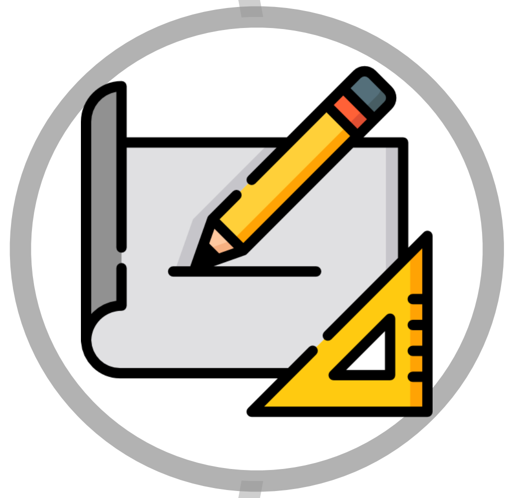
Dedicated, Tightly Integrated Static Pipelines	**Internet / Early generation Datacenters** ✓	**Today's Datacenters us-latency, 100s of Gbps** ✗

Today's network stacks are on the brink of breakdown

Rearchitecture is inevitable

This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware



NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

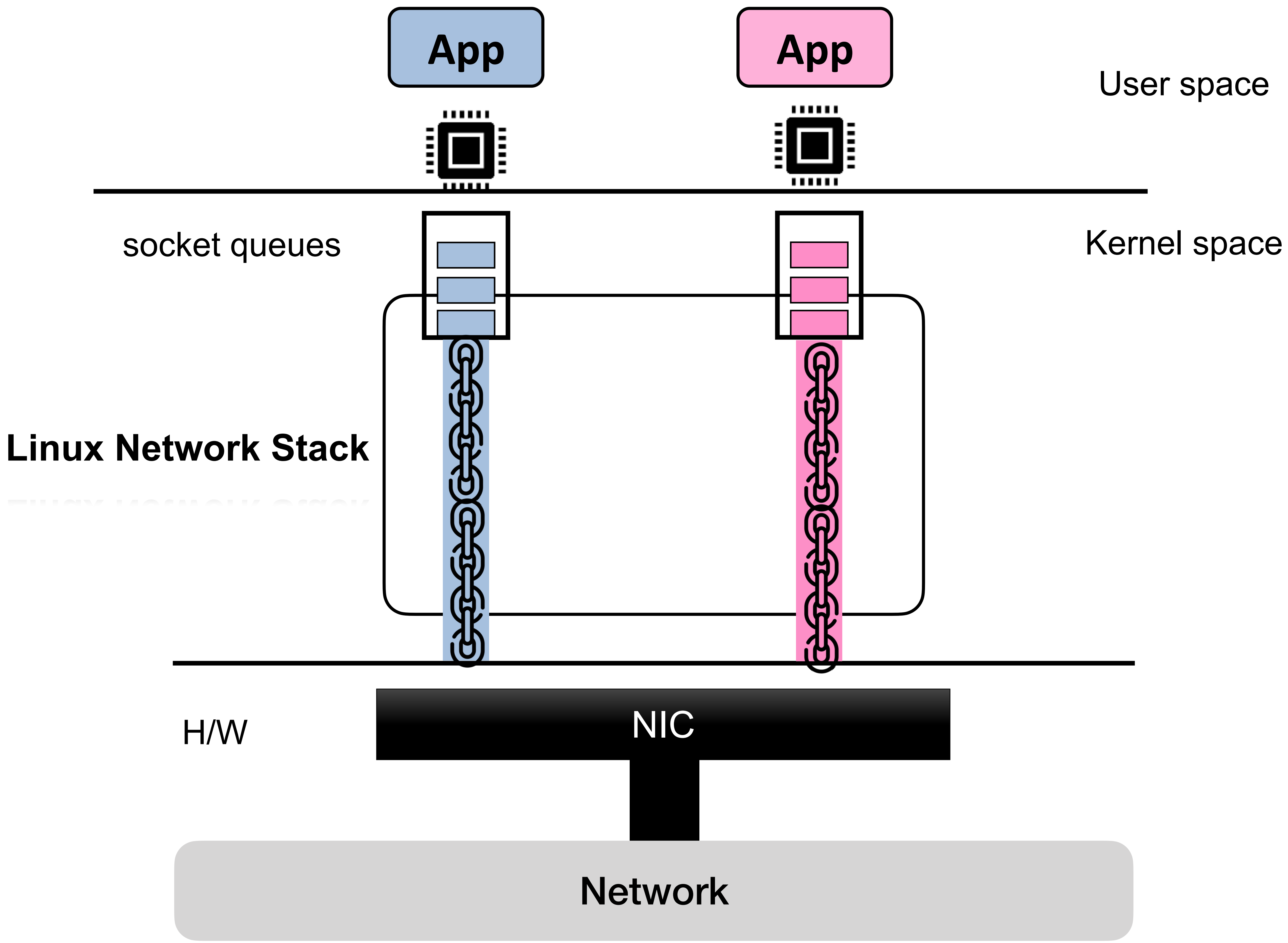
Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

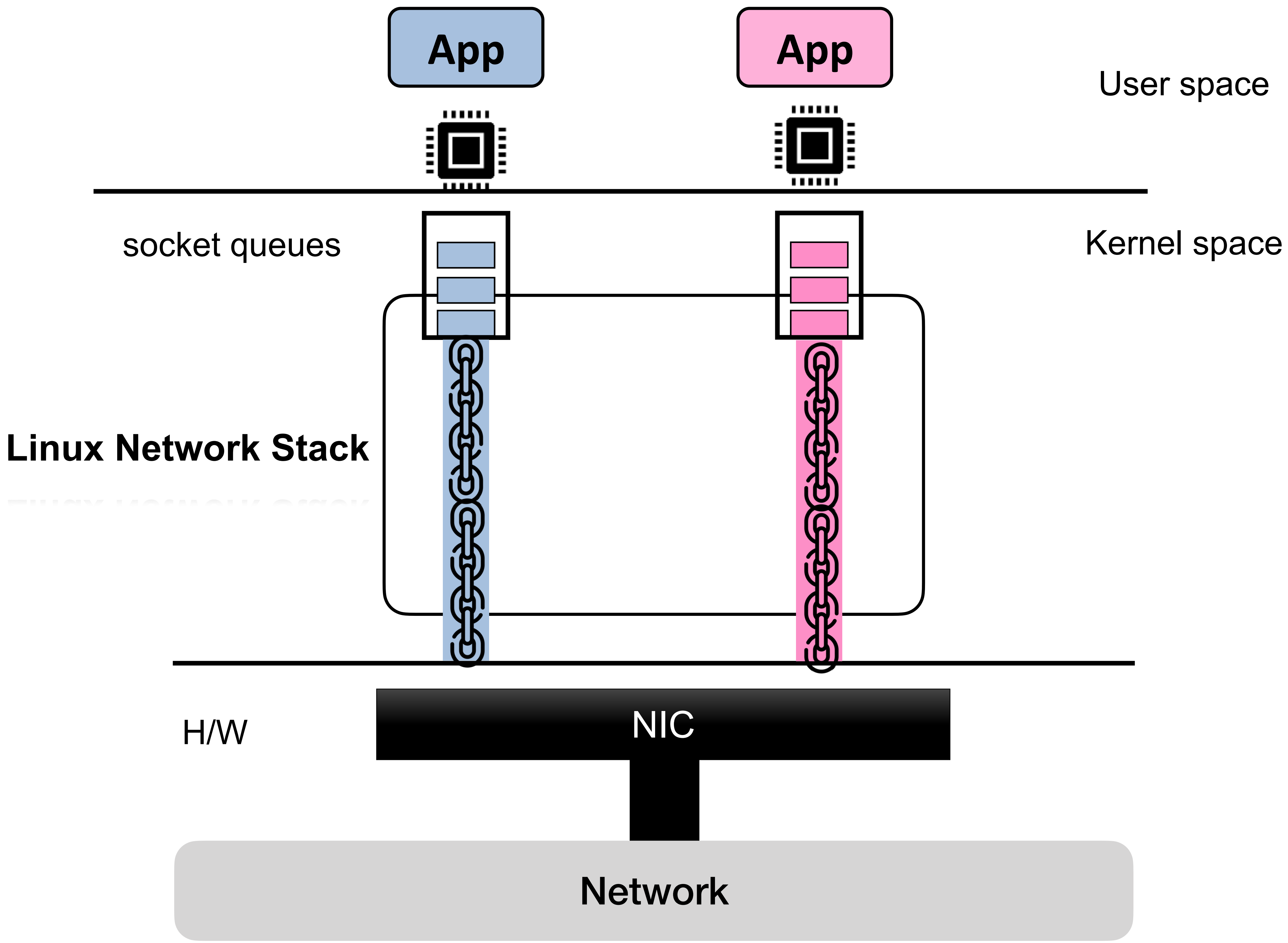
Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers



Disaggregating the Host Network Stack

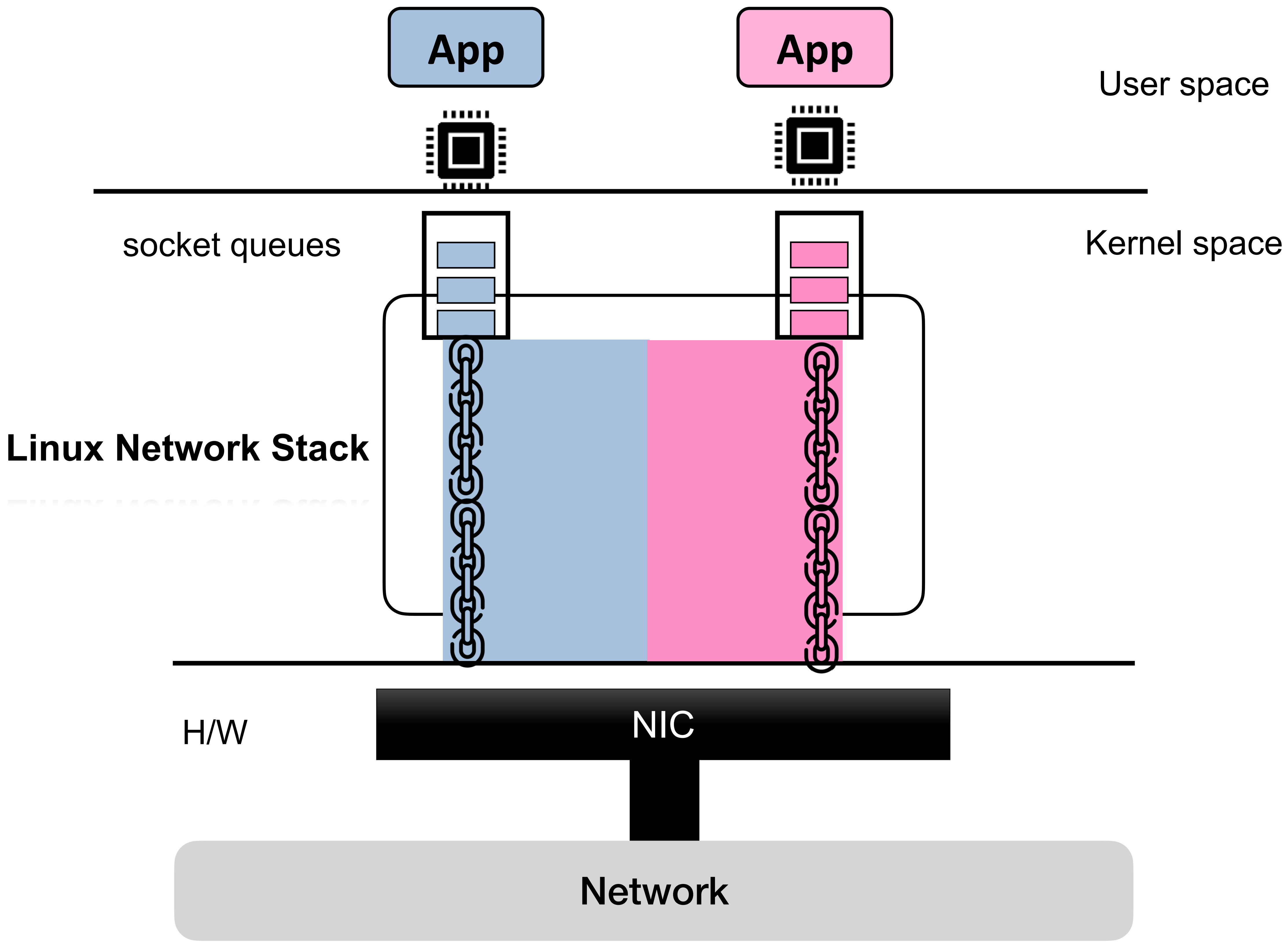
Key Idea: Disaggregate packet processing pipeline into separate layers



~~Dedicated~~ **Shared**
Shared resources across pipes

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

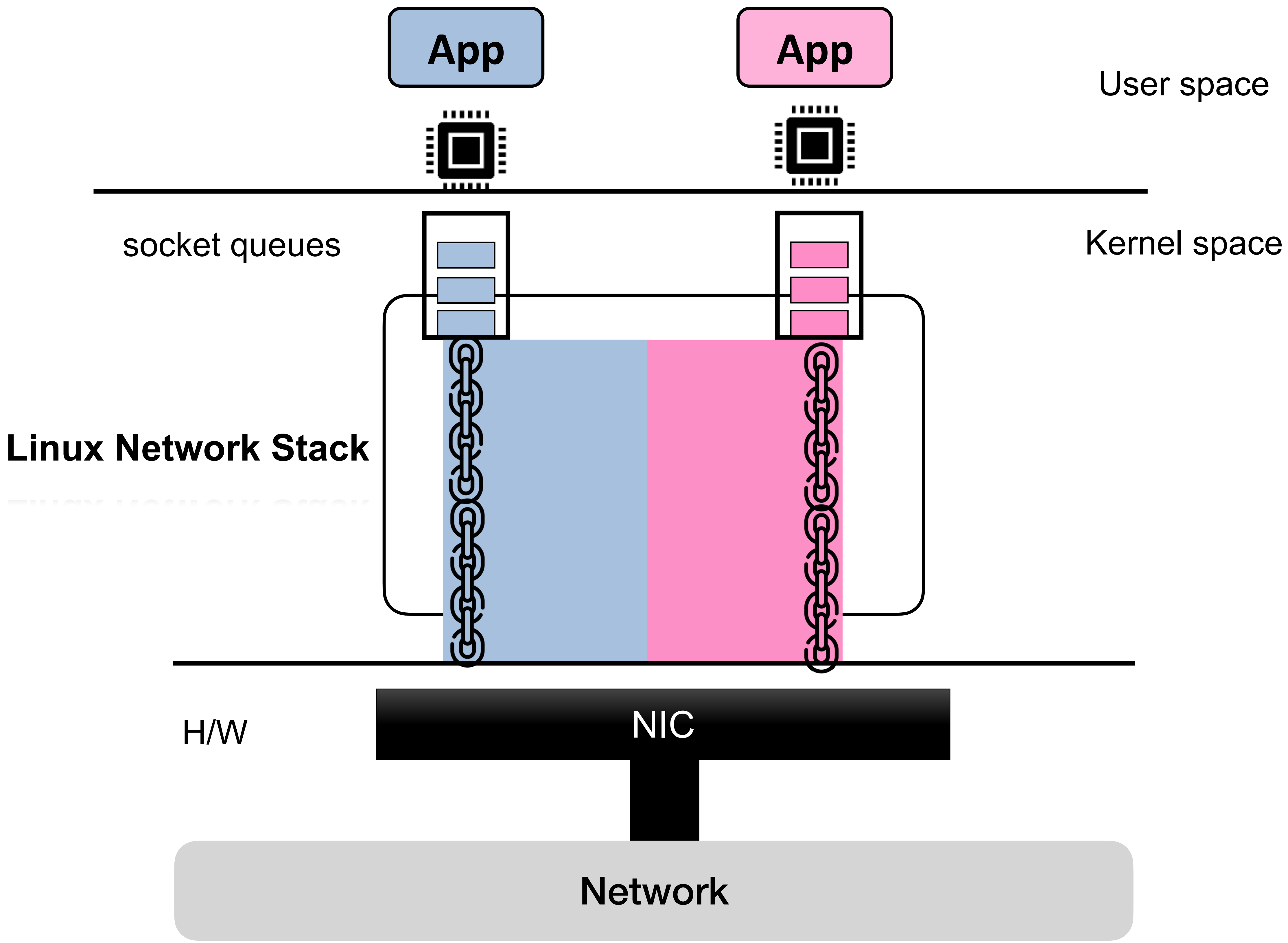


~~Dedicated~~ **Shared**

Shared resources across pipes

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

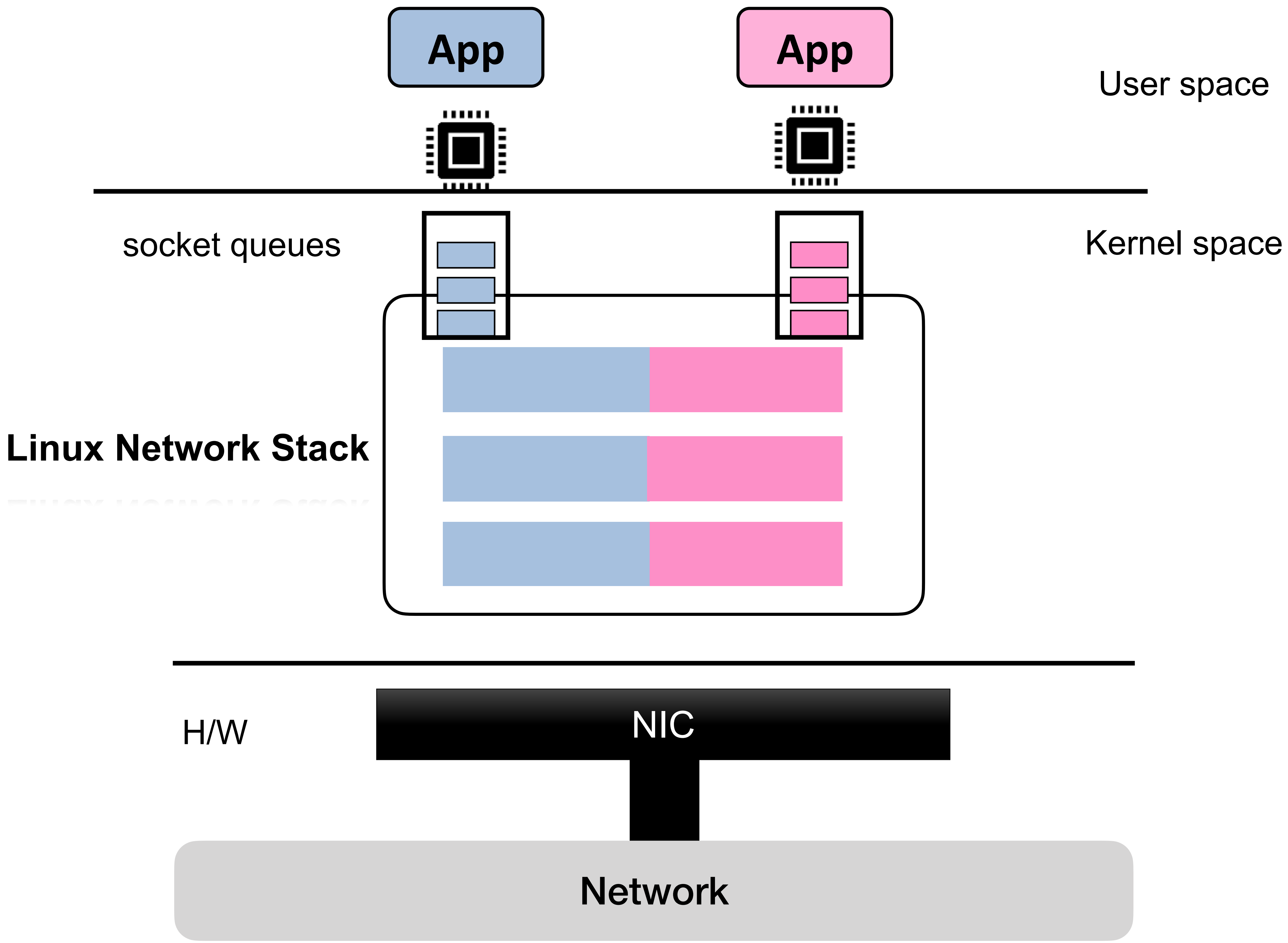


~~Dedicated~~ **Shared**
Shared resources across pipes

~~Tightly Integrated~~ **Loosely Coupled**
Different parts of pipeline are decoupled

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers

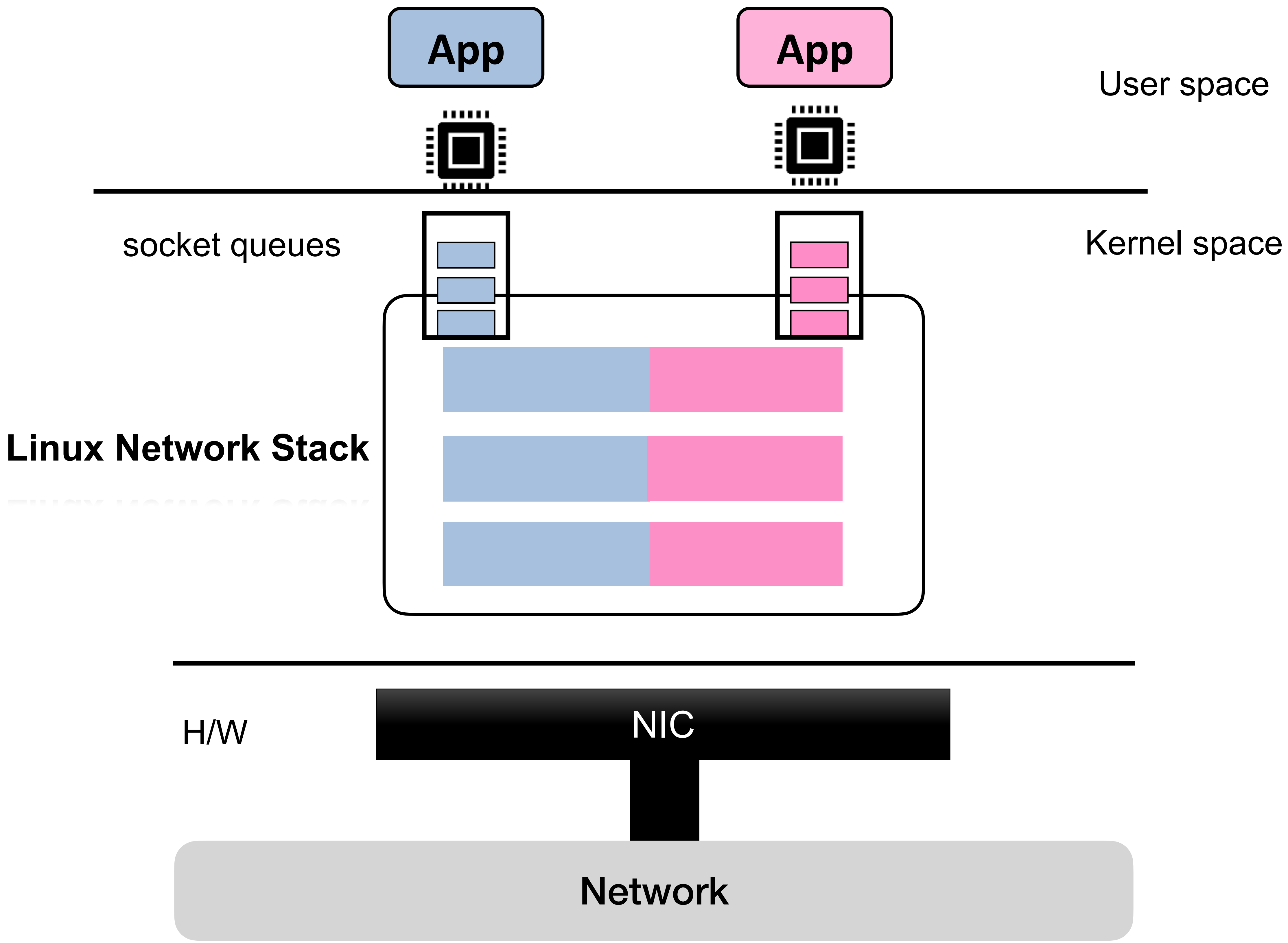


~~Dedicated~~ **Shared**
Shared resources across pipes

~~Tightly Integrated~~ **Loosely Coupled**
Different parts of pipeline are decoupled

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers



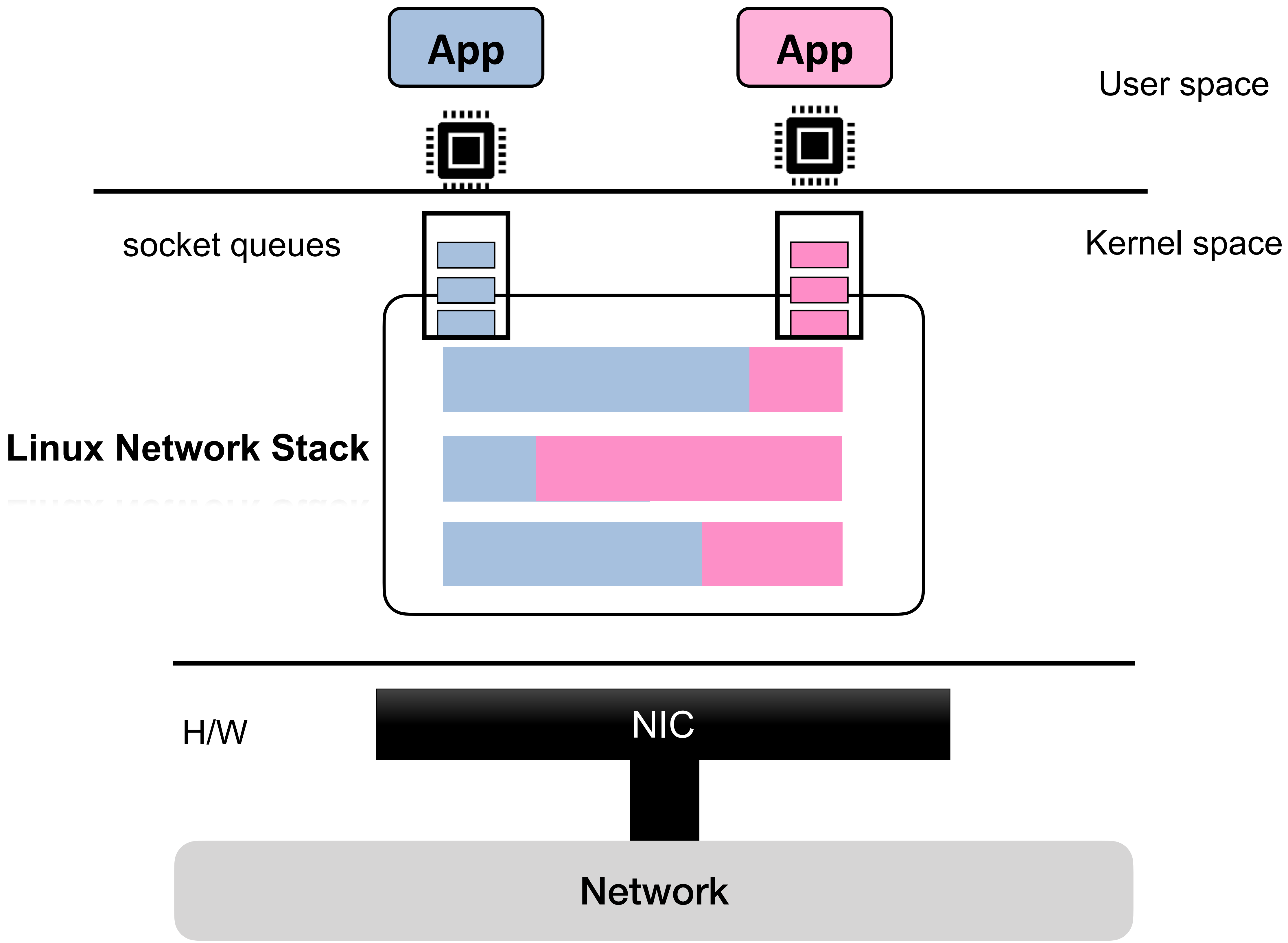
~~Dedicated~~ **Shared**
Shared resources across pipes

~~Tightly Integrated~~ **Loosely Coupled**
Different parts of pipeline are decoupled

~~Static~~ **Dynamic**
Independent scaling of resources allocated to different parts of the pipeline
(Based on resource availability and other pipes)

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers



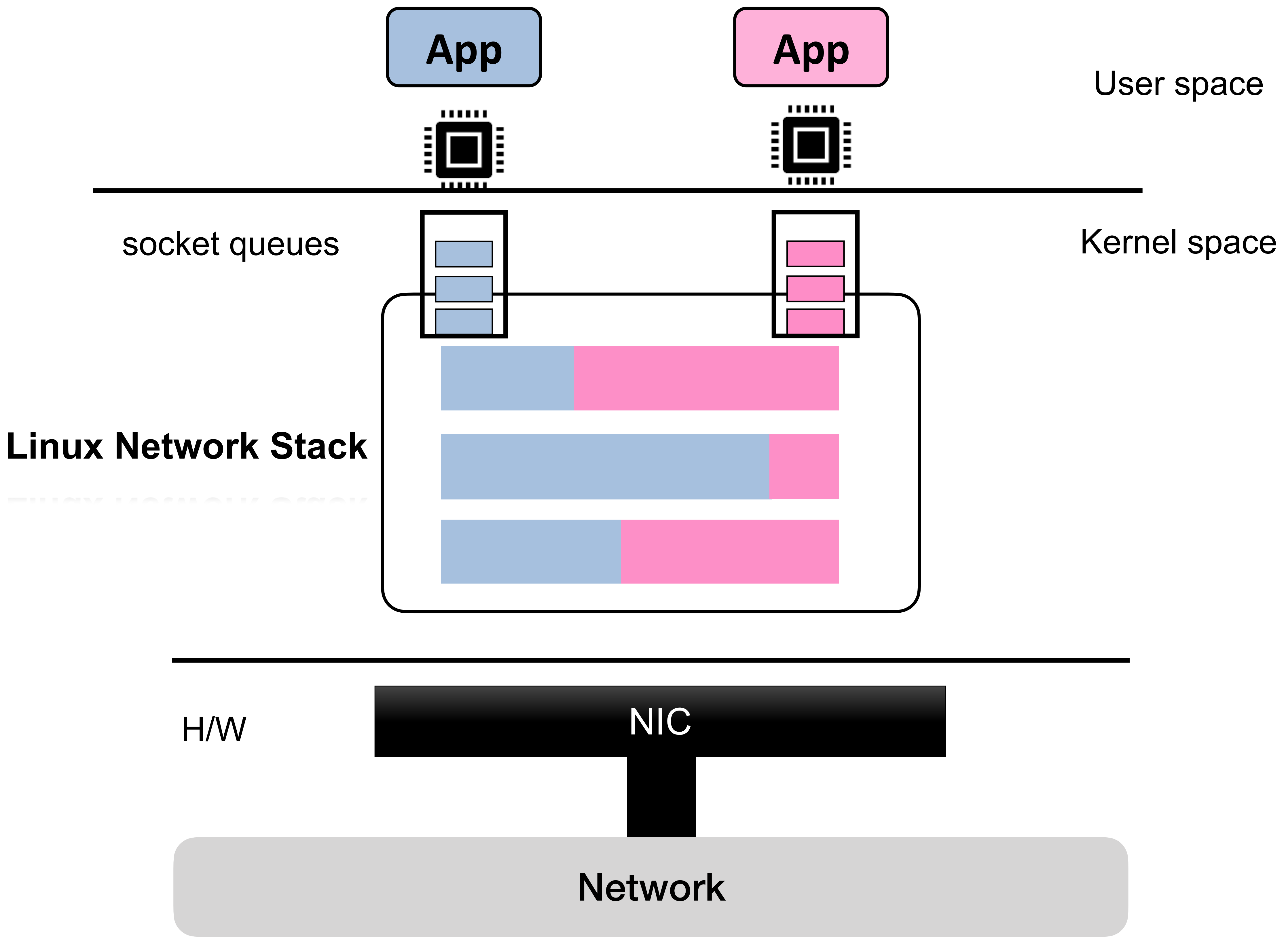
~~Dedicated~~ **Shared**
Shared resources across pipes

~~Tightly Integrated~~ **Loosely Coupled**
Different parts of pipeline are decoupled

~~Static~~ **Dynamic**
Independent scaling of resources allocated to different parts of the pipeline
(Based on resource availability and other pipes)

Disaggregating the Host Network Stack

Key Idea: Disaggregate packet processing pipeline into separate layers



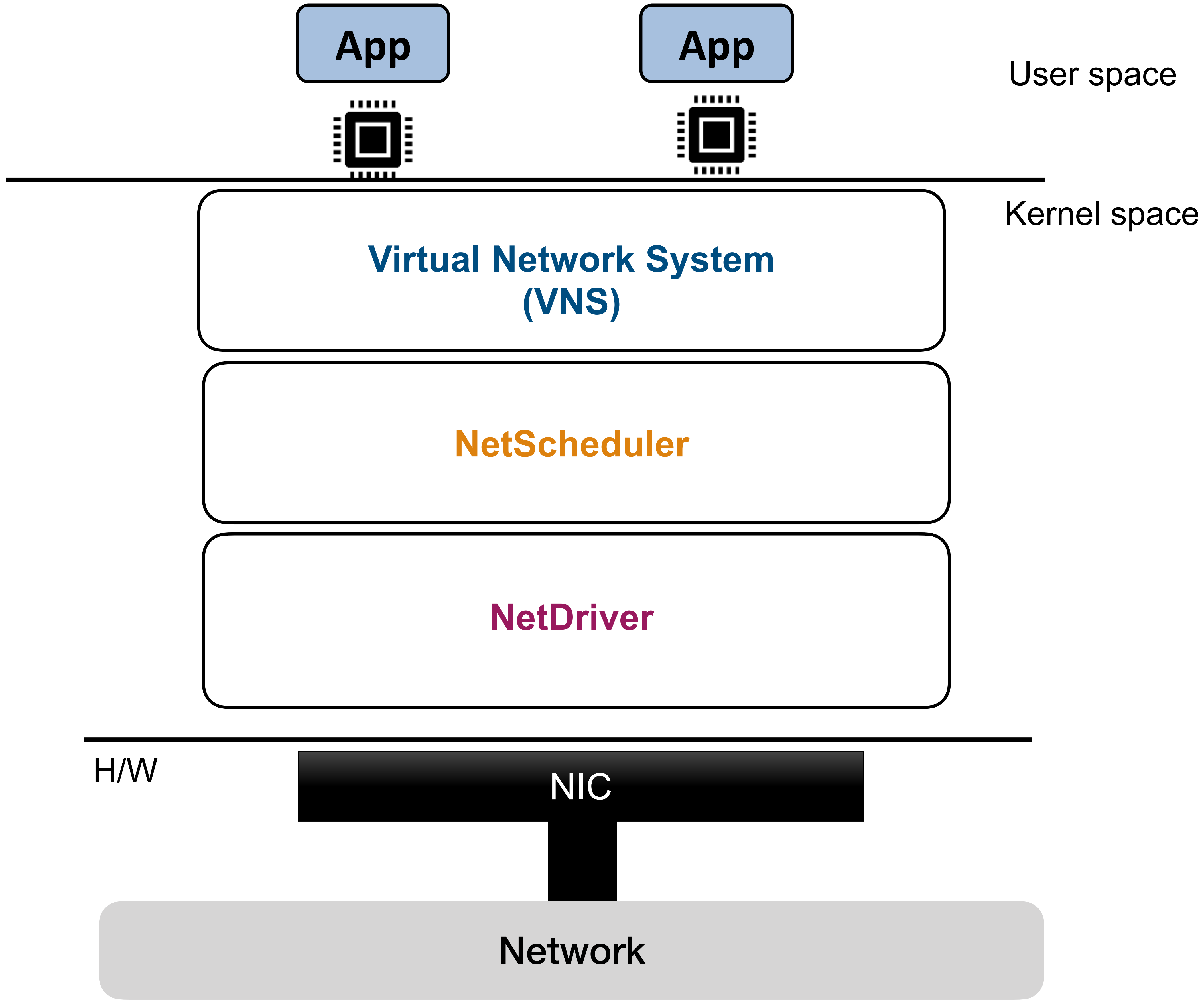
~~Dedicated~~ **Shared**
Shared resources across pipes

~~Tightly Integrated~~ **Loosely Coupled**
Different parts of pipeline are decoupled

~~Static~~ **Dynamic**
Independent scaling of resources allocated to different parts of the pipeline
(Based on resource availability and other pipes)

NetChannel Architecture

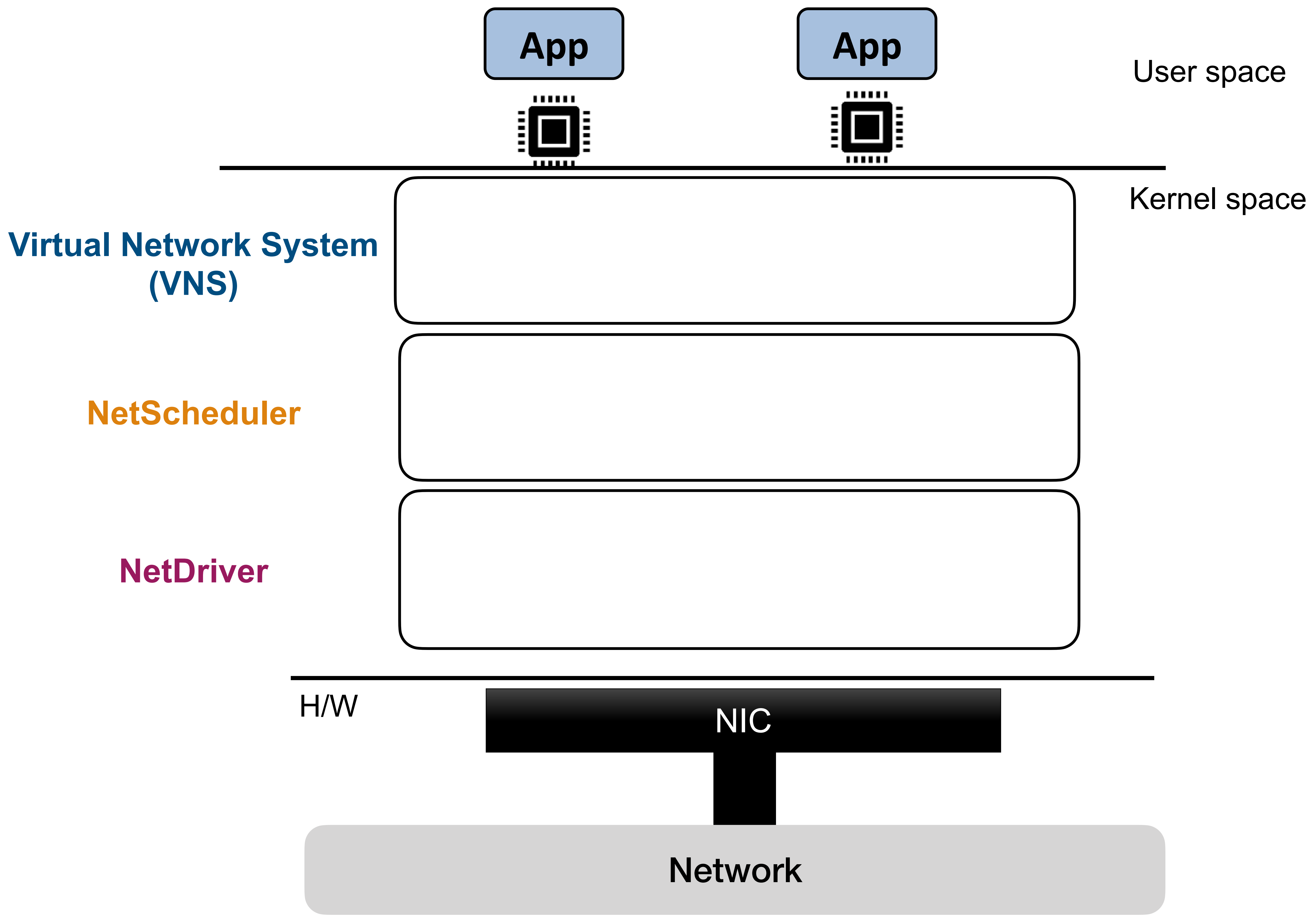
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



Icon made by Freepik from www.flaticon.com

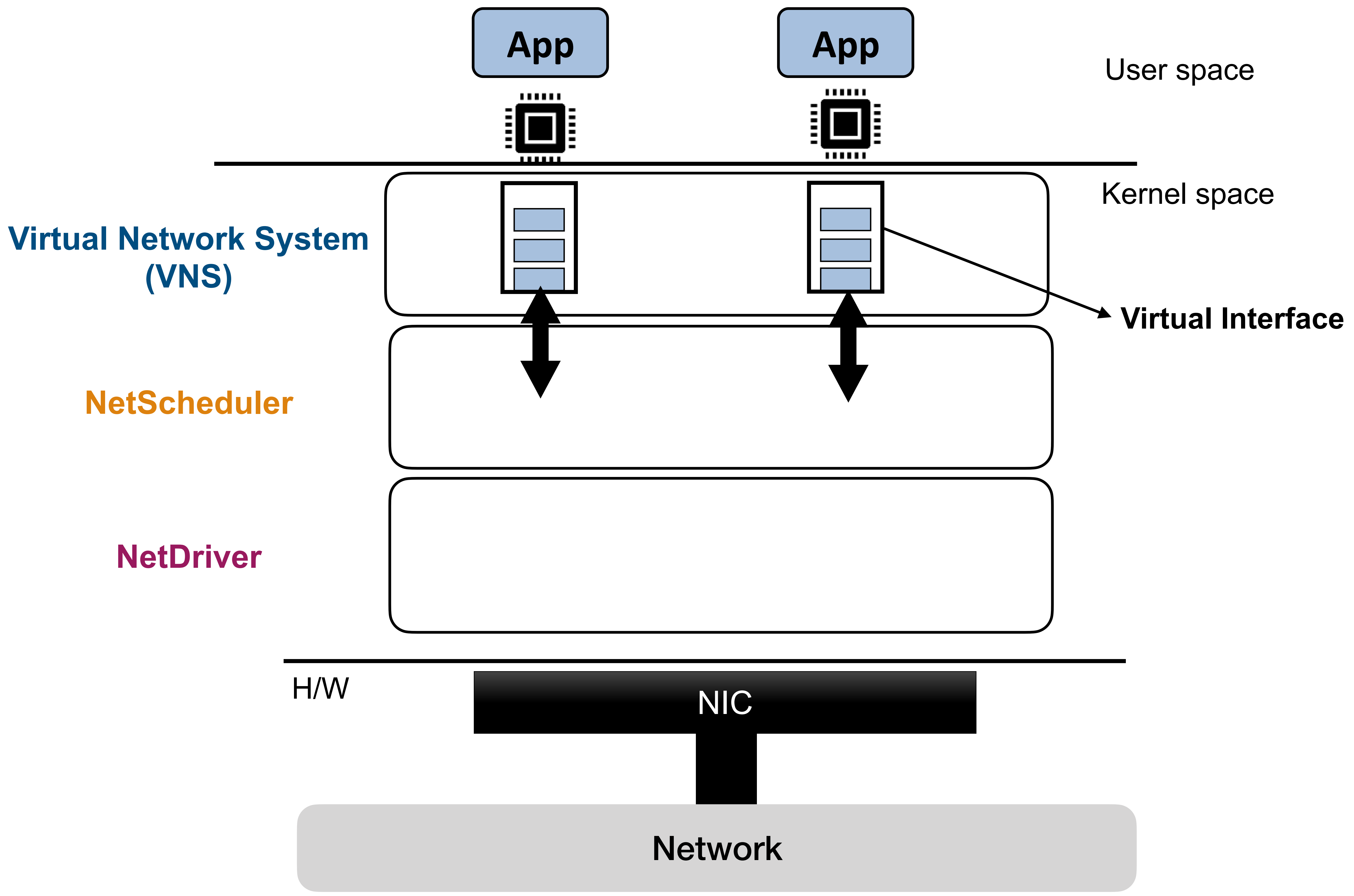
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



NetChannel Architecture

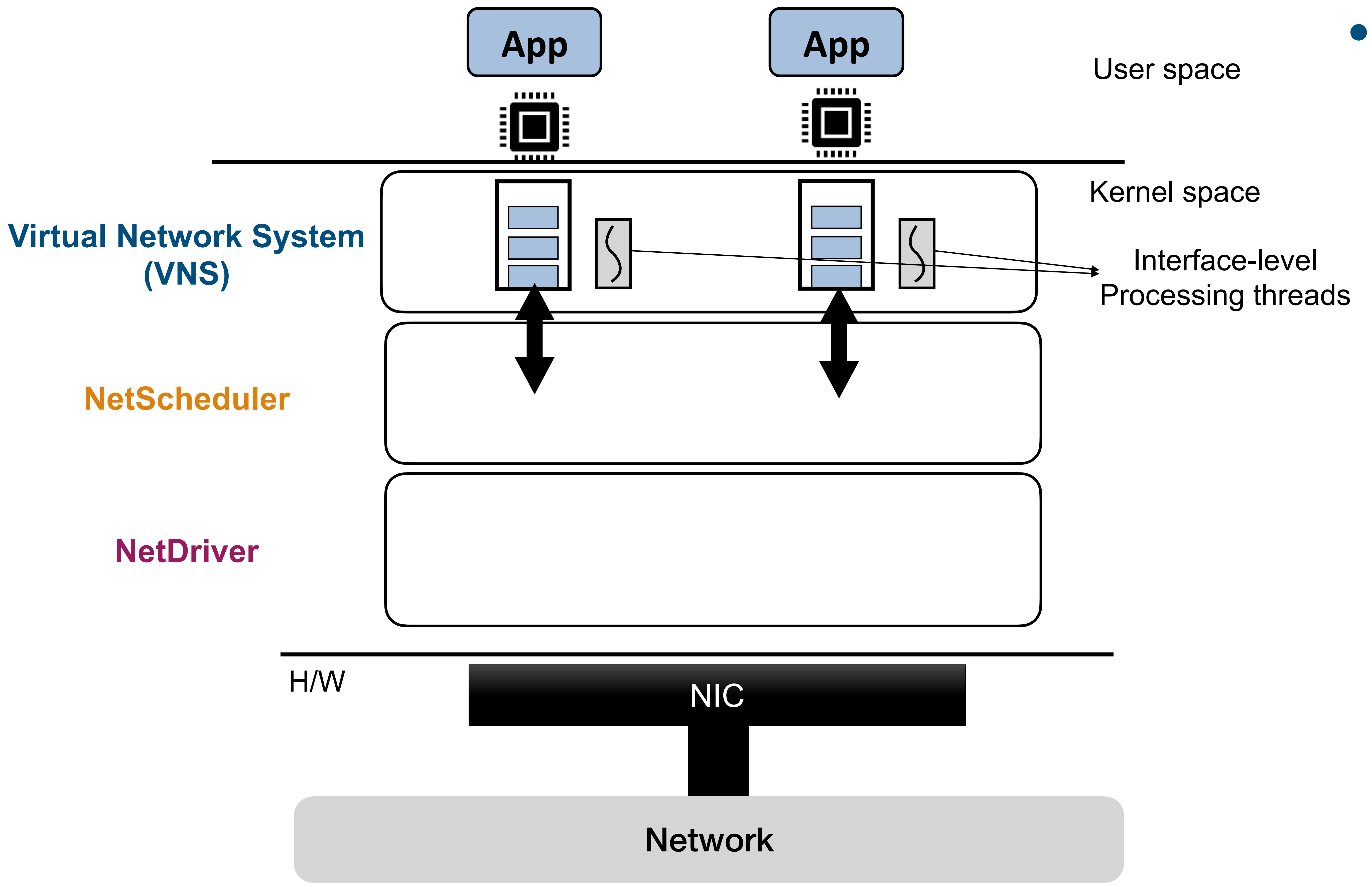
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



- **Virtual Network System (VNS)**
  - Provide **virtual** interfaces (e.g, socket, RPC, ..)

NetChannel Architecture

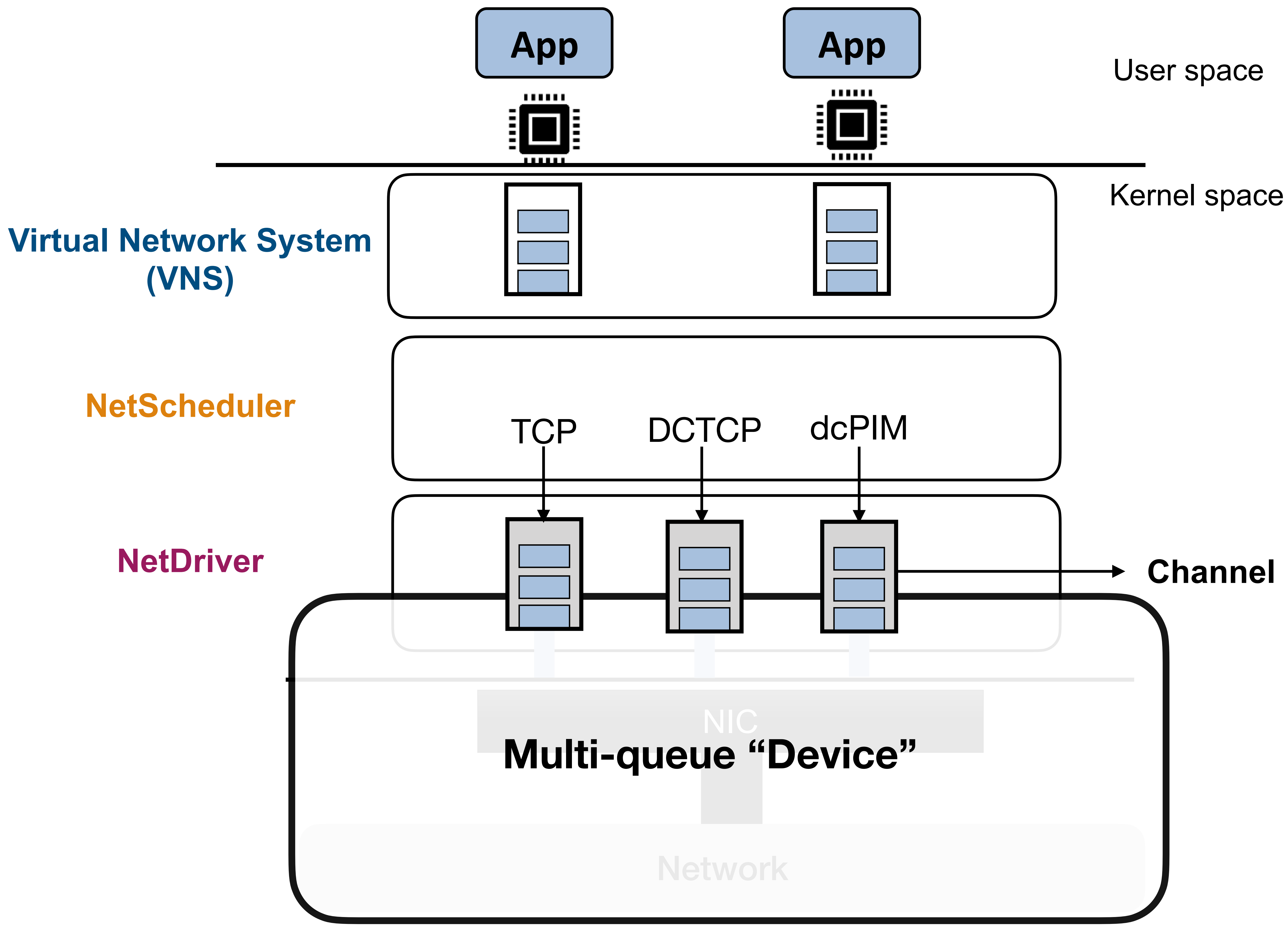
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



- **Virtual Network System (VNS)**
  - Provide **virtual** interfaces (e.g, socket, RPC, ..)
  - Decouples Interface-level Processing

NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers

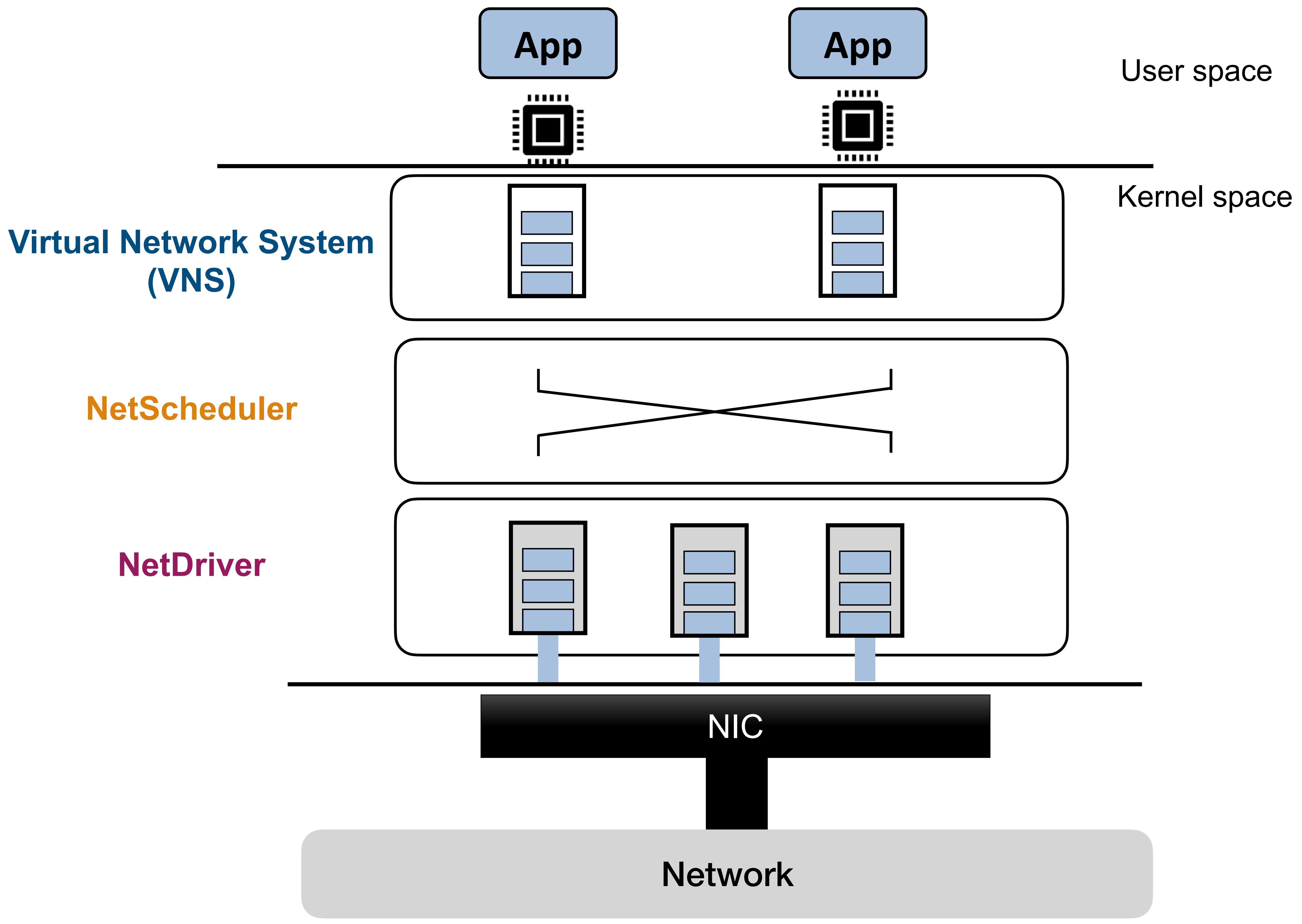


- **Virtual Network System (VNS)**
  - Provide **virtual** interfaces (e.g, socket interface)
  - Decouples Interface-level Processing
- **NetDriver**
  - Abstract the network as a multi-queue "device"
    - through **channel** abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing

Icon made by Freepik from www.flaticon.com

NetChannel Architecture

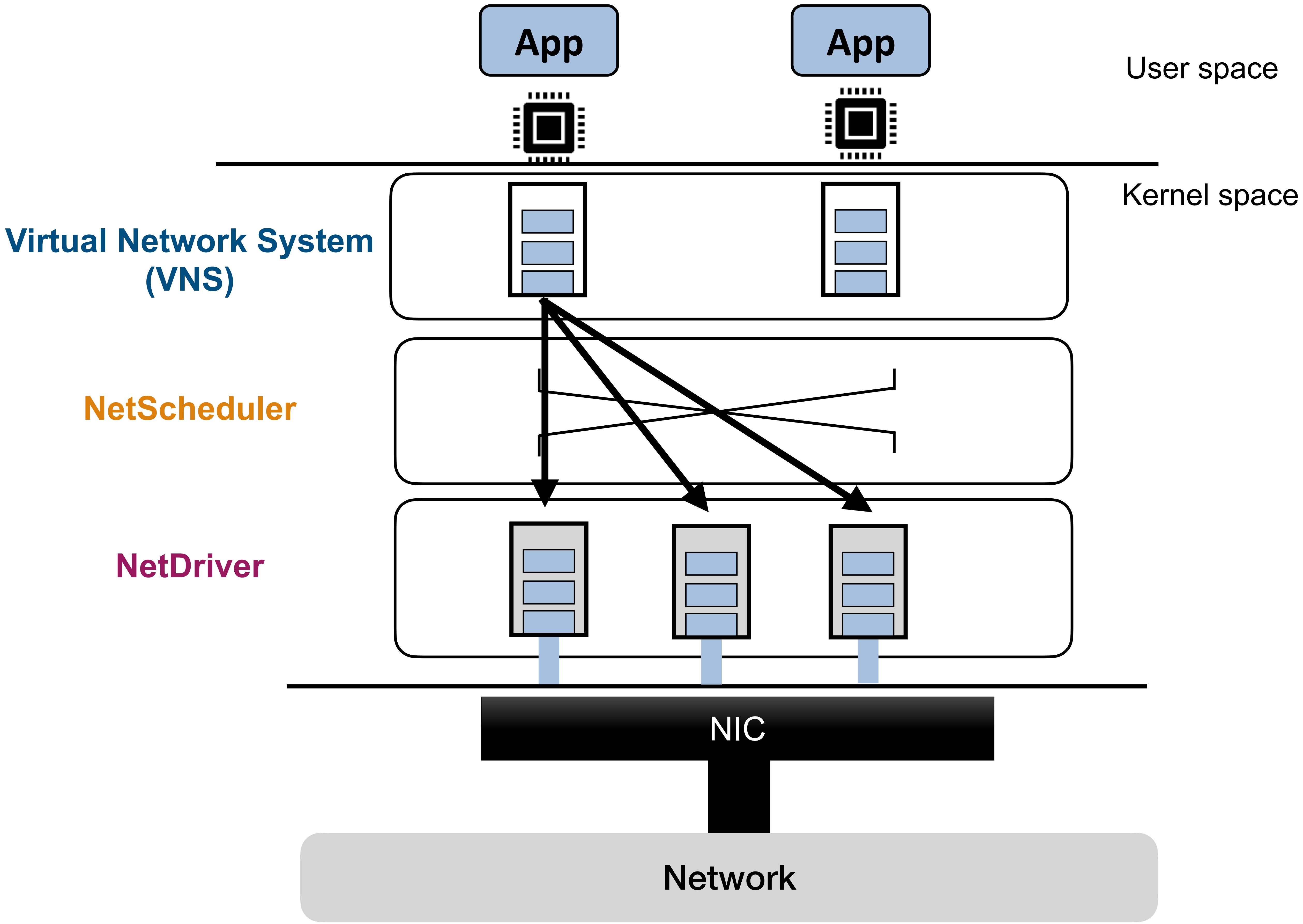
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



- **Virtual Network System (VNS)**
  - Provide **virtual** interfaces (e.g, socket interface)
  - Enables decoupling Interface-level Processing
- **NetScheduler**
  - Multiplex/Demultiplex data
    - between virtual interfaces and channels
  - Enables flexible scheduling policies
- **NetDriver**
  - Abstract the network as a multi-queue "device"
    - through **channel** abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing

NetChannel Architecture

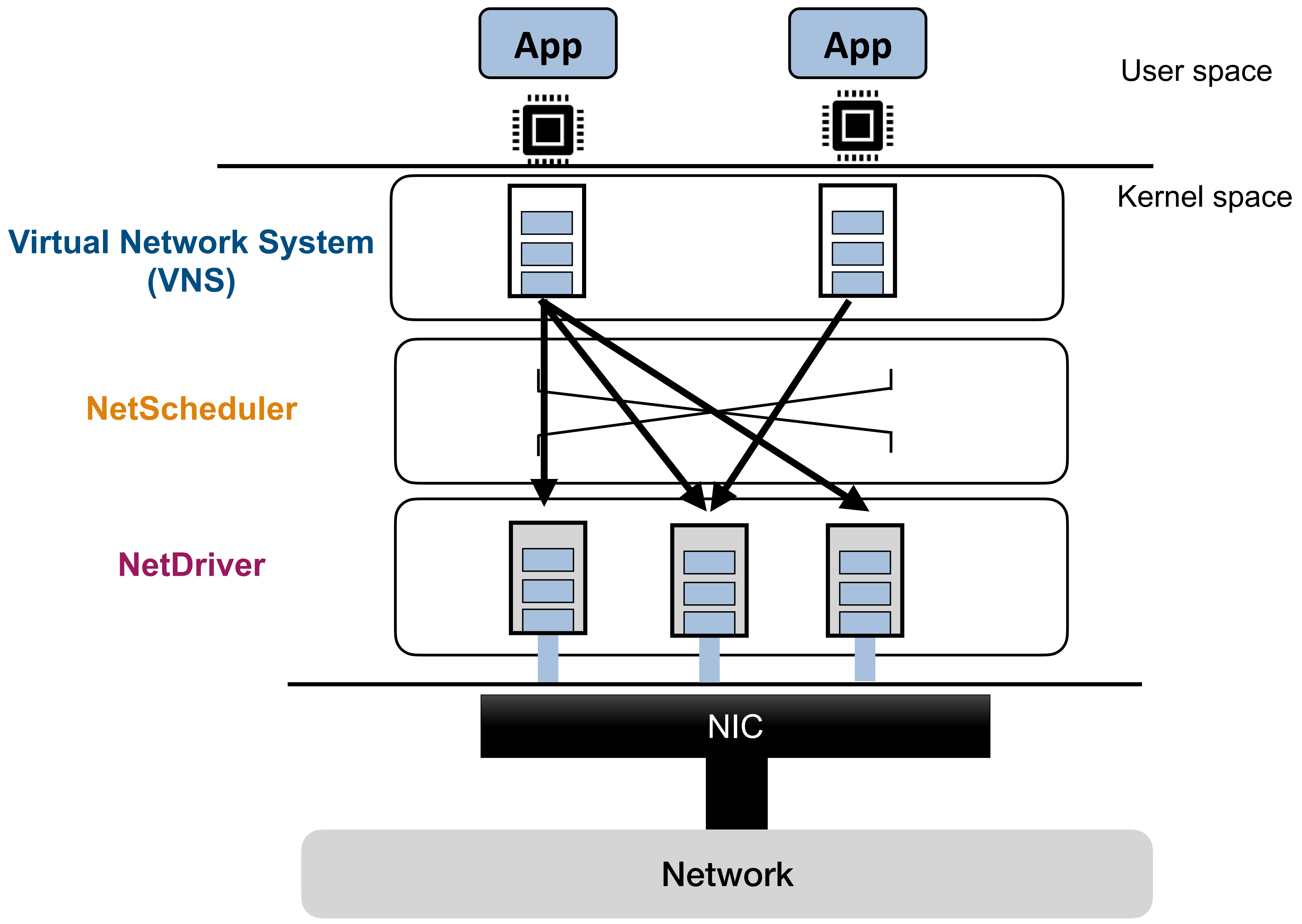
NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



- **Virtual Network System (VNS)**
  - Provide **virtual** interfaces (e.g, socket interface)
  - Enables decoupling Interface-level Processing
- **NetScheduler**
  - Multiplex/Demultiplex data
    - between virtual interfaces and channels
  - Enables flexible scheduling policies
- **NetDriver**
  - Abstract the network as a multi-queue “device”
    - through **channel** abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing

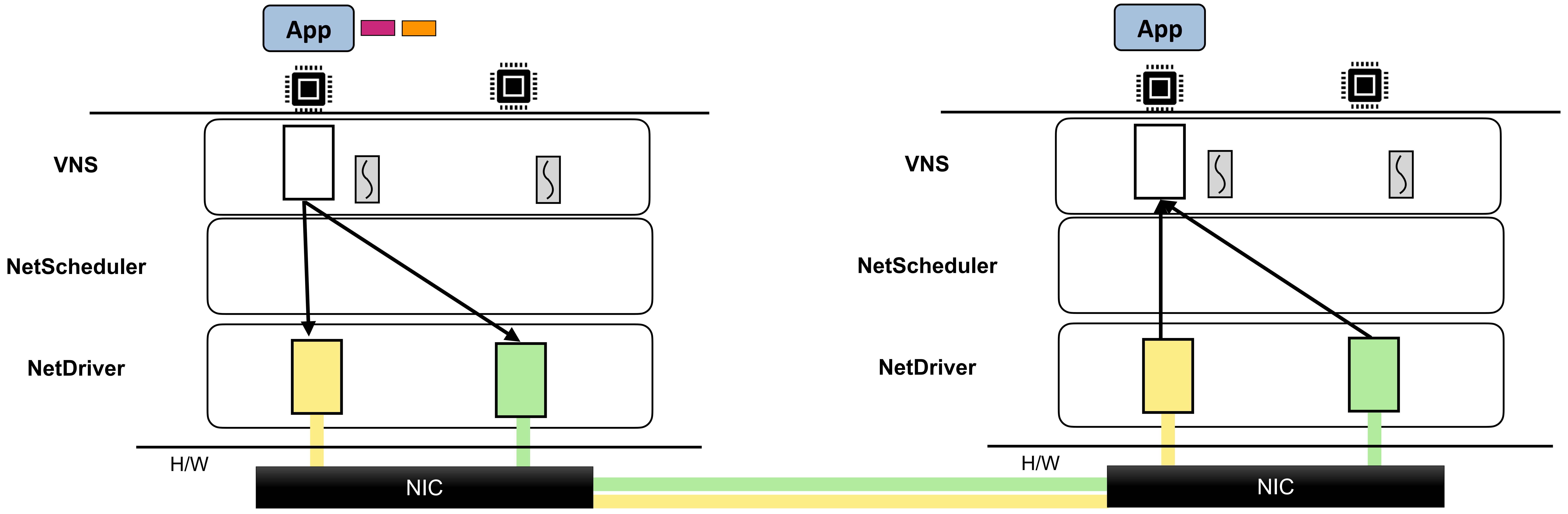
NetChannel Architecture

NetChannel disaggregates the Host Network Stack into 3 loosely-coupled layers



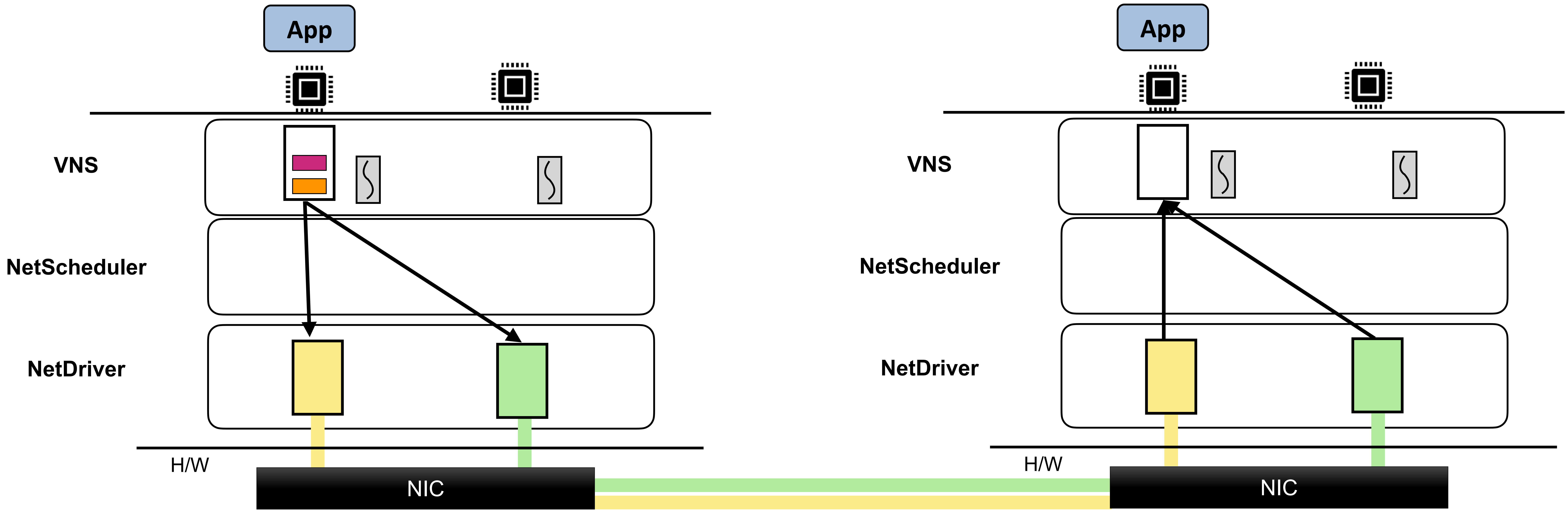
- **Virtual Network System (VNS)**
  - Provide **virtual** interfaces (e.g, socket interface)
  - Enables decoupling Interface-level Processing
- **NetScheduler**
  - Multiplex/Demultiplex data
    - between virtual interfaces and channels
  - Enables flexible scheduling policies
- **NetDriver**
  - Abstract the network as a multi-queue "device"
    - through **channel** abstraction
  - Enables easy integration of new transports
  - Decouples Network Layer Processing

NetChannel End-to-End Operation



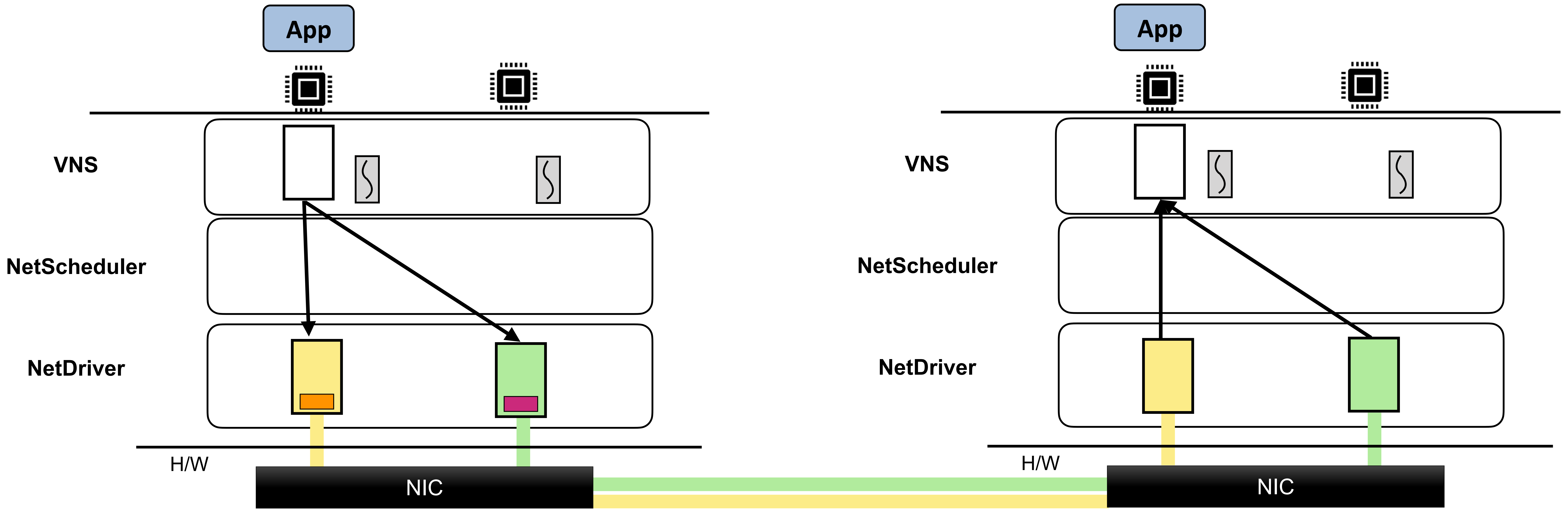
(Note: One can define different NetScheduler policies)

NetChannel End-to-End Operation



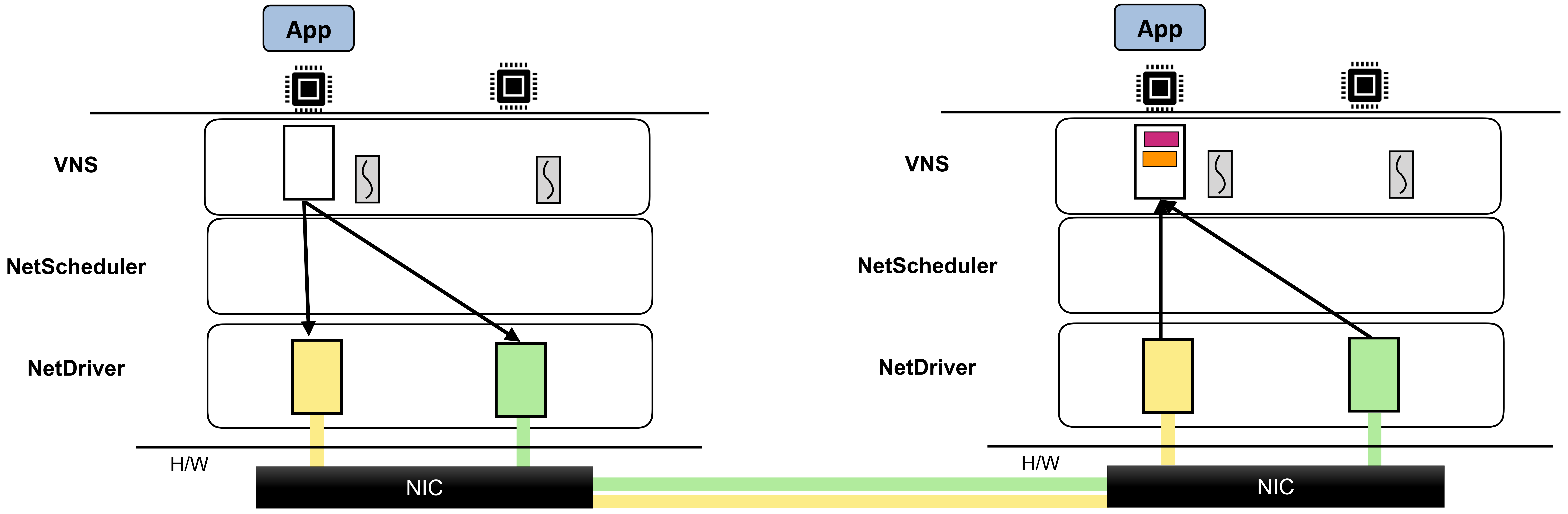
(Note: One can define different NetScheduler policies)

NetChannel End-to-End Operation



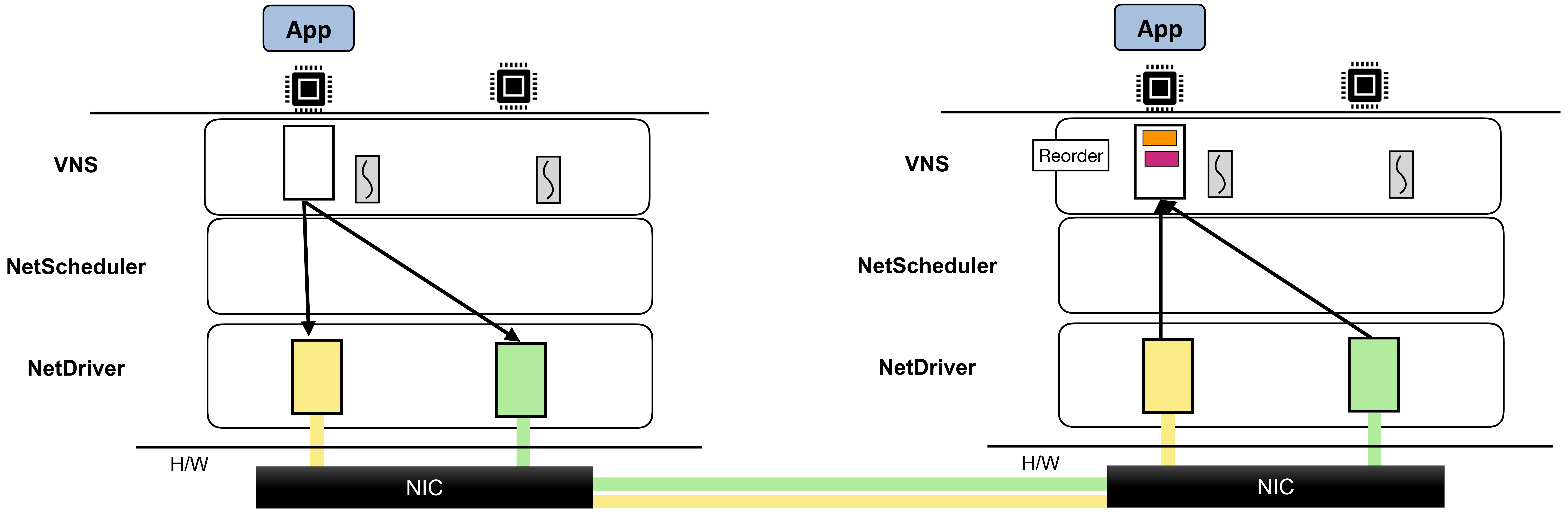
(Note: One can define different NetScheduler policies)

NetChannel End-to-End Operation



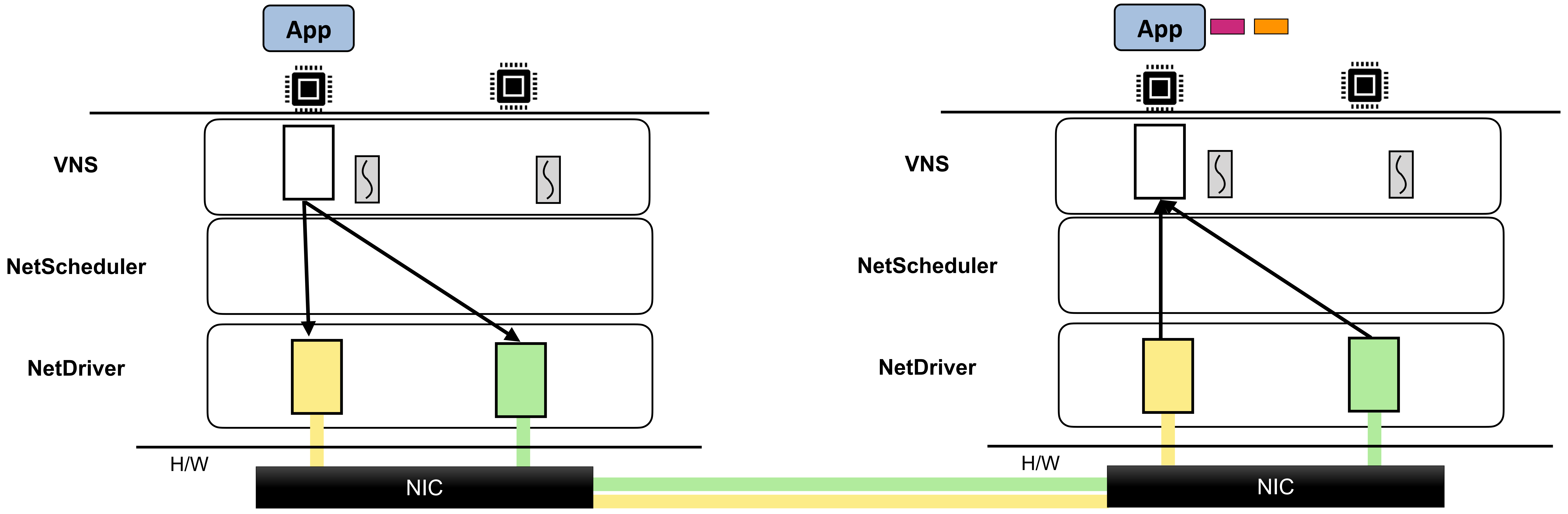
(Note: One can define different NetScheduler policies)

NetChannel End-to-End Operation



(Note: One can define different NetScheduler policies)

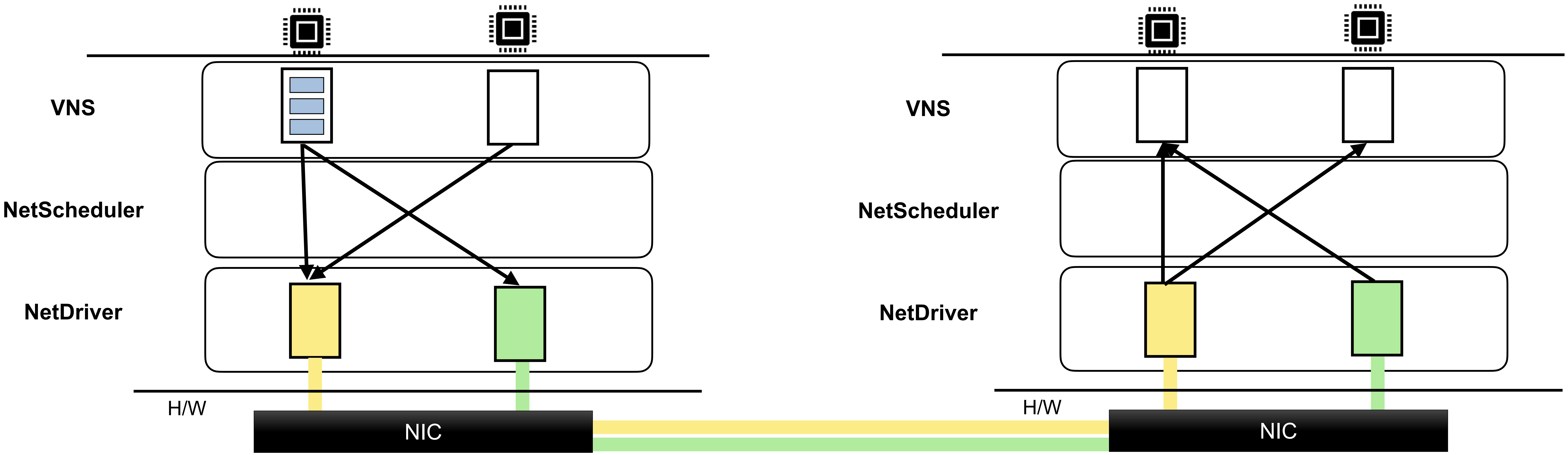
NetChannel End-to-End Operation



(Note: One can define different NetScheduler policies)

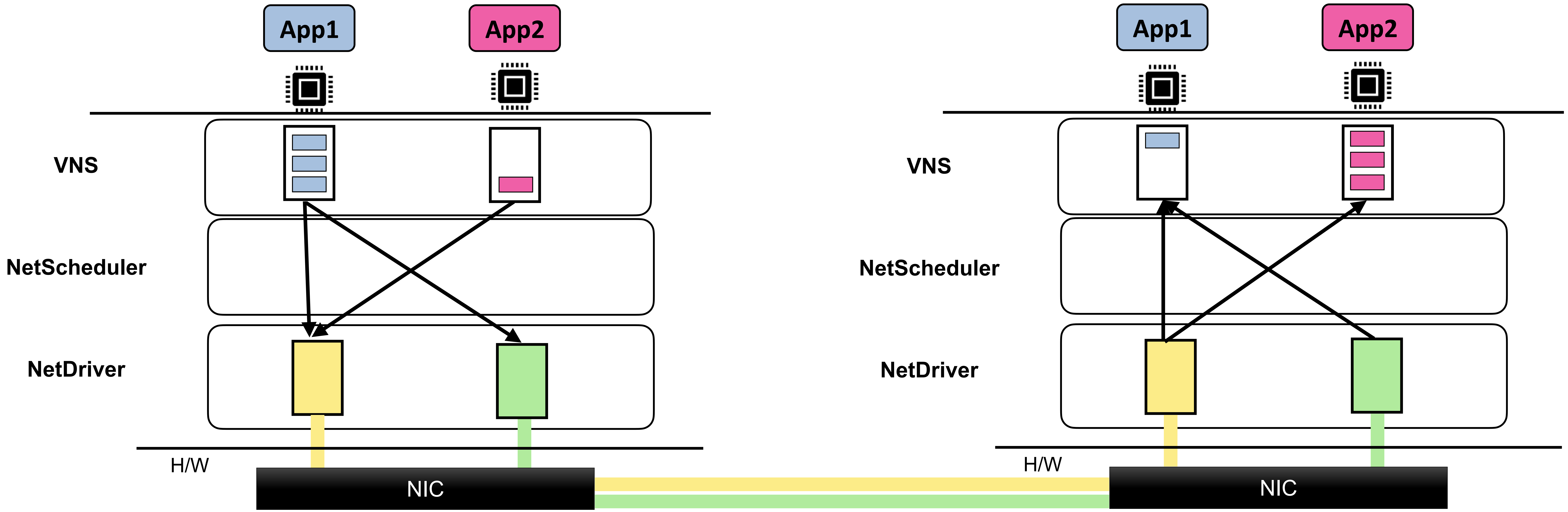
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



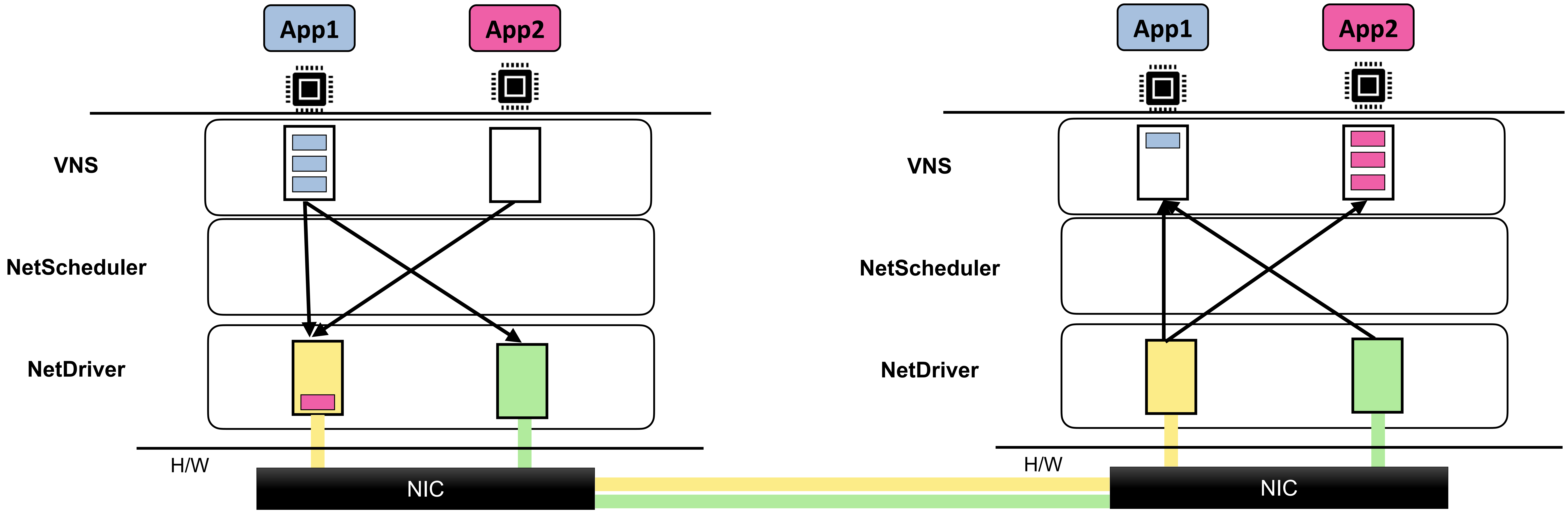
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



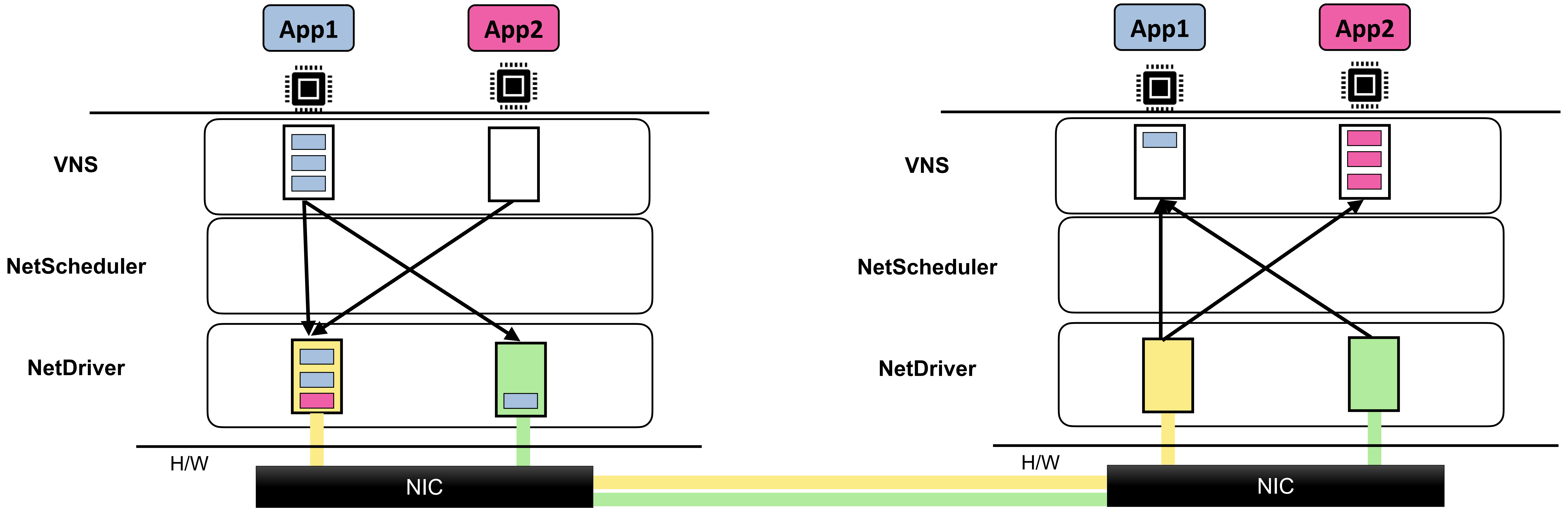
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



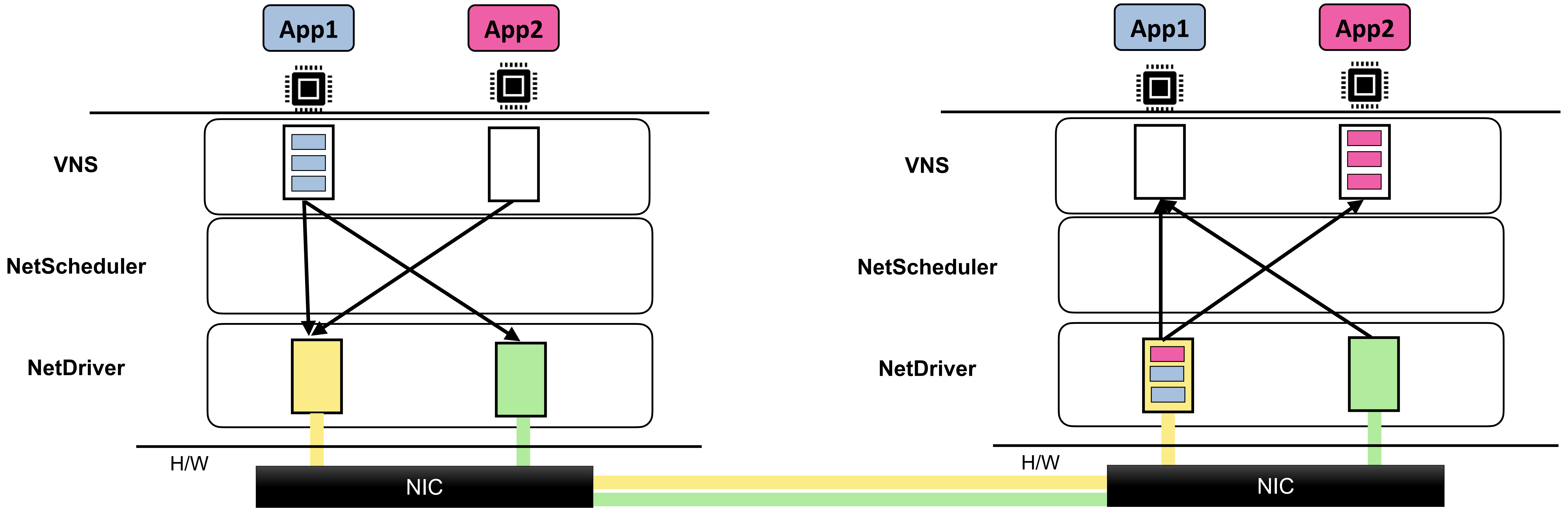
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



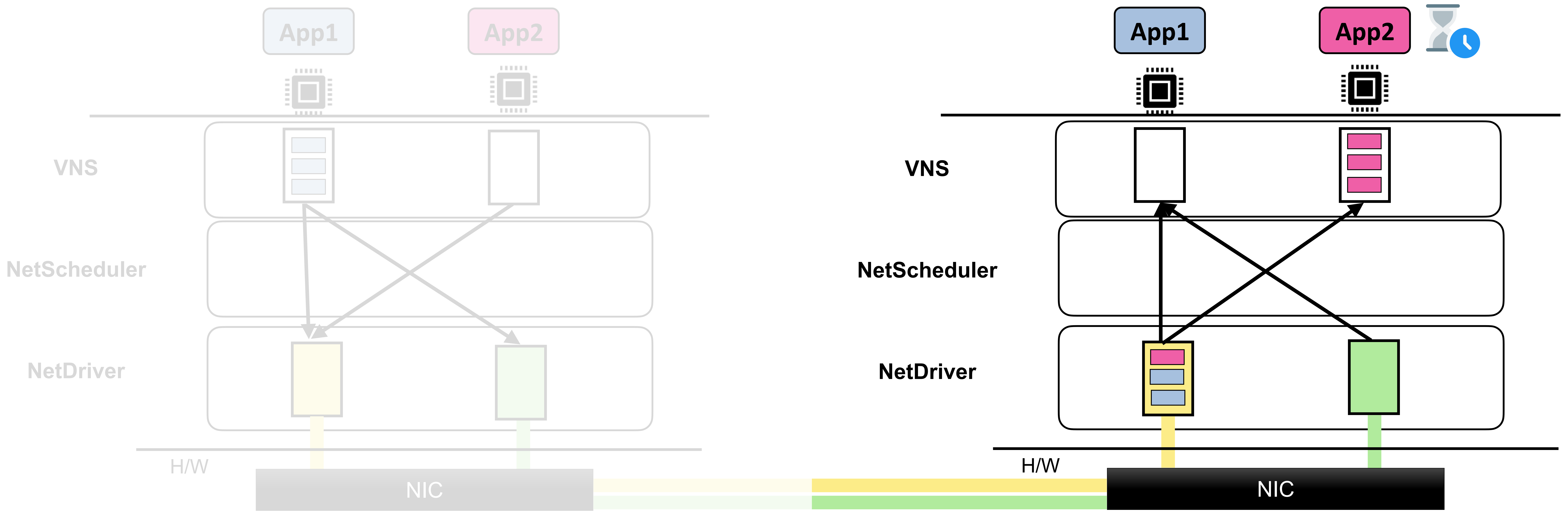
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



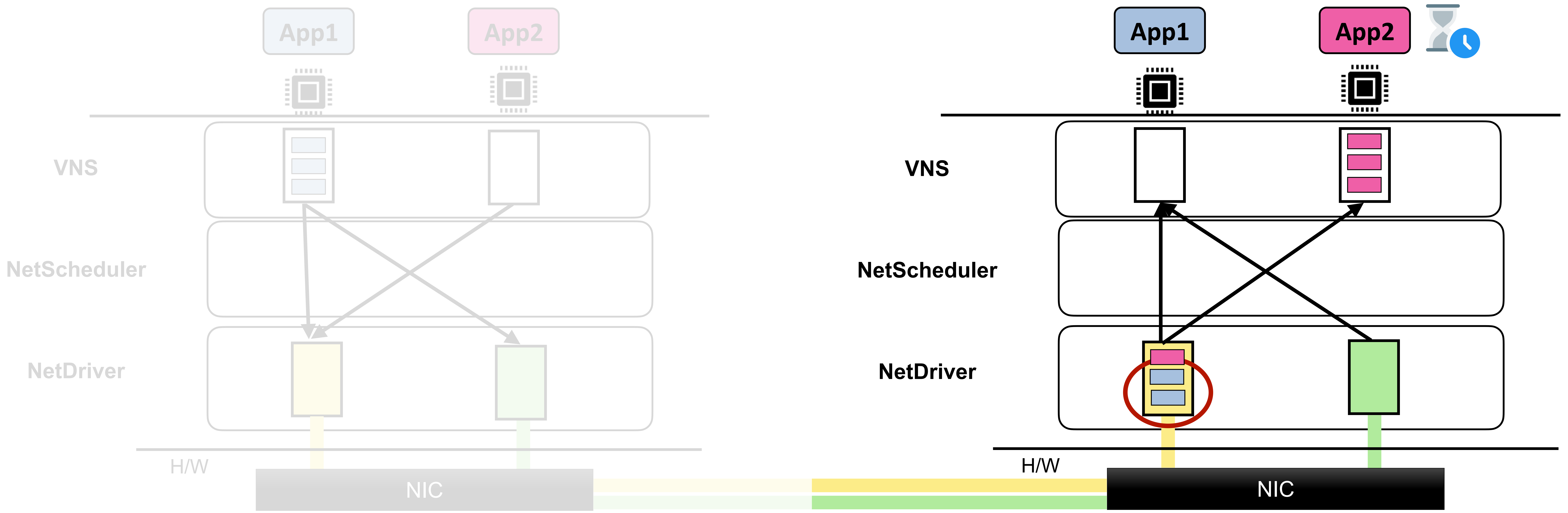
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



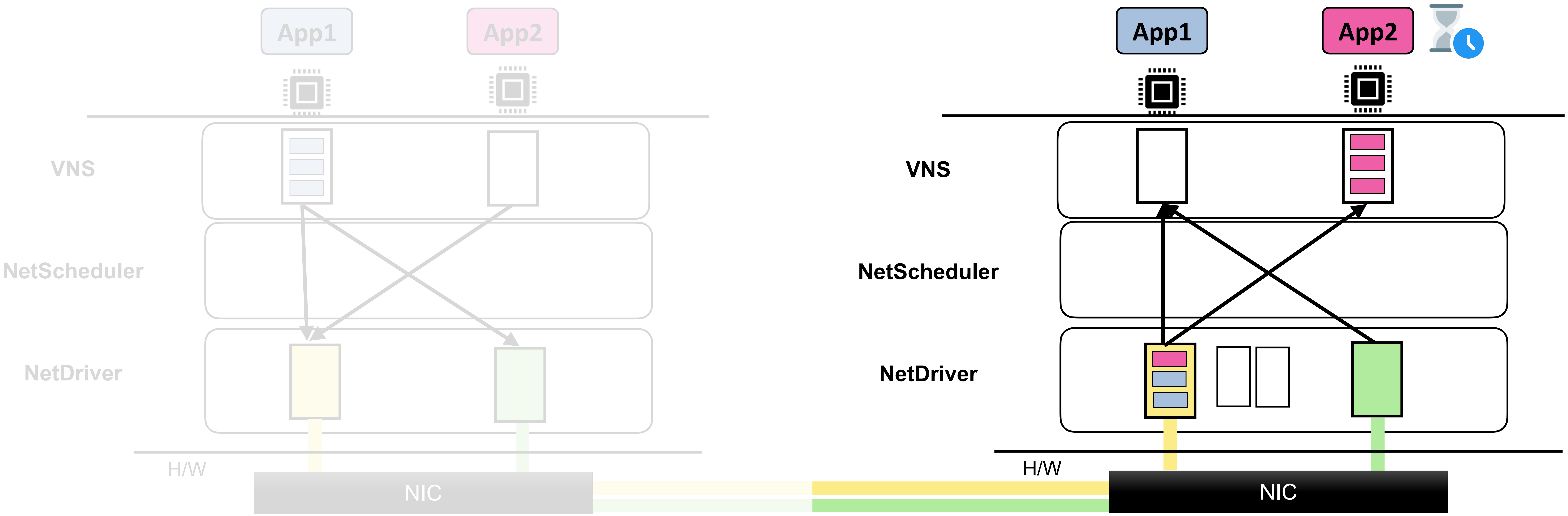
NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



NetChannel End-to-End Operation

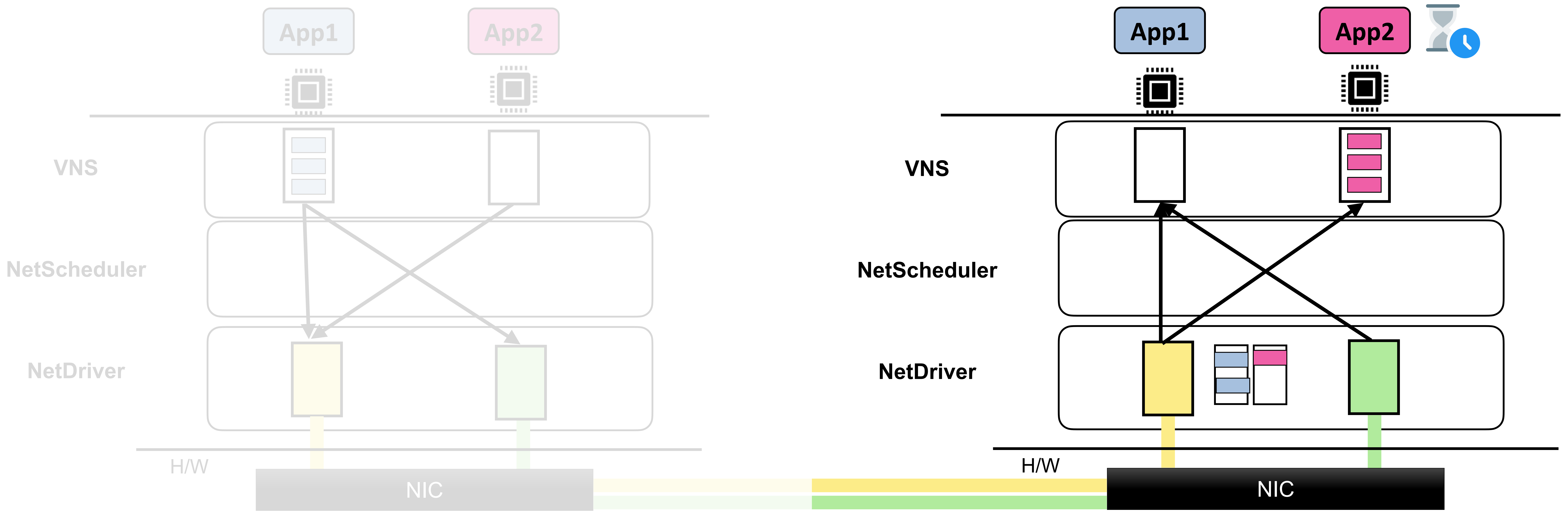
Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



Per-virtual socket response queues associated with each channel

NetChannel End-to-End Operation

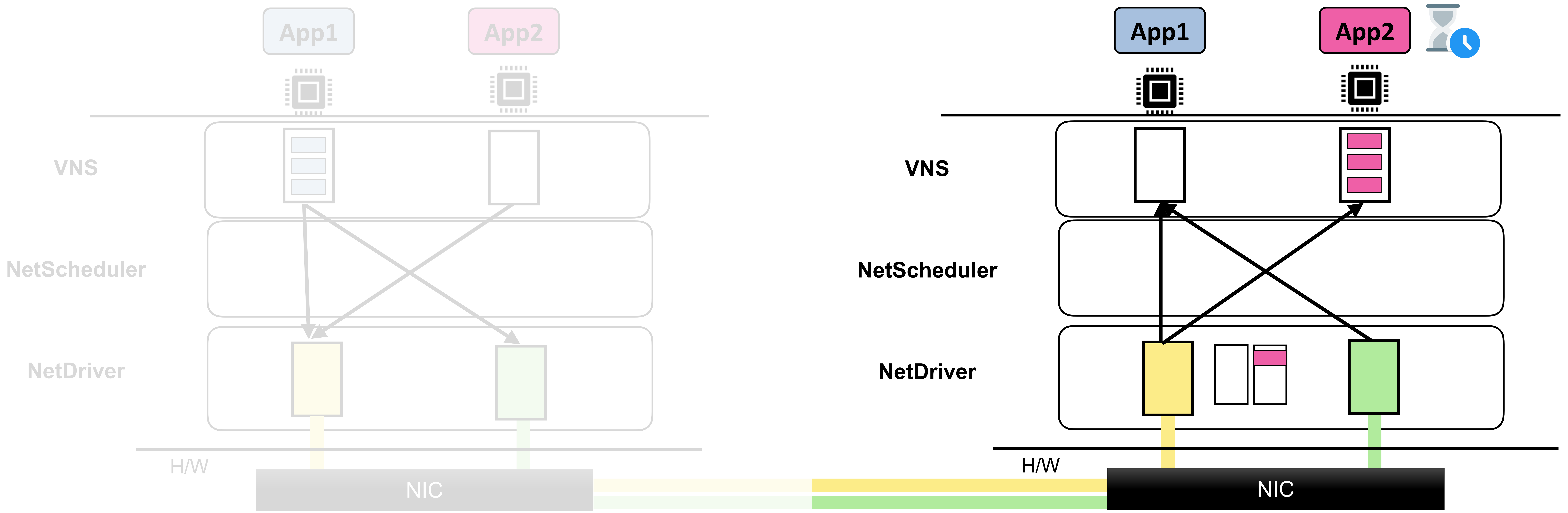
Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



Per-virtual socket response queues associated with each channel

NetChannel End-to-End Operation

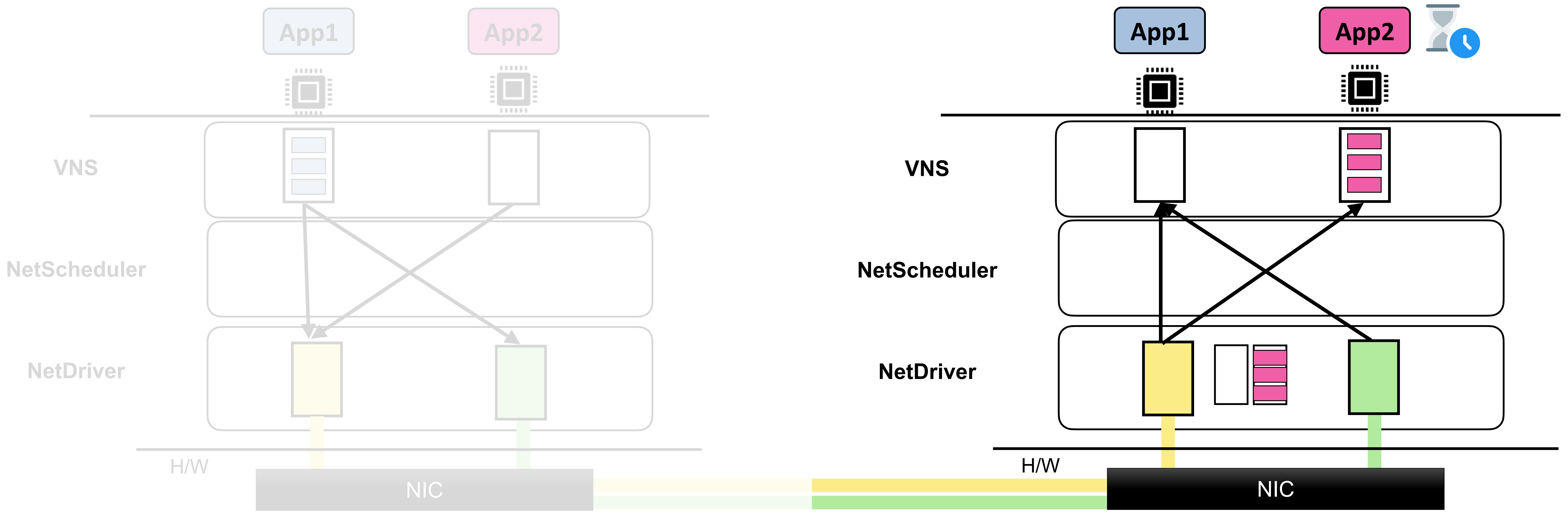
Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



Per-virtual socket response queues associated with each channel

NetChannel End-to-End Operation

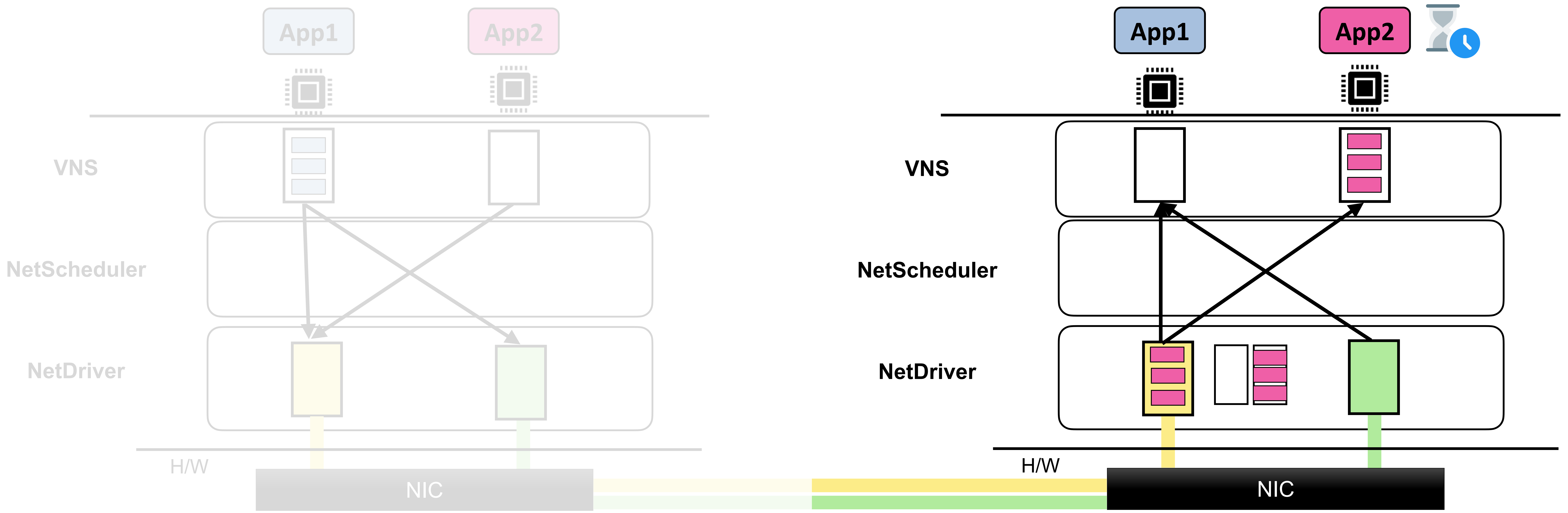
Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



Per-virtual socket response queues associated with each channel

NetChannel End-to-End Operation

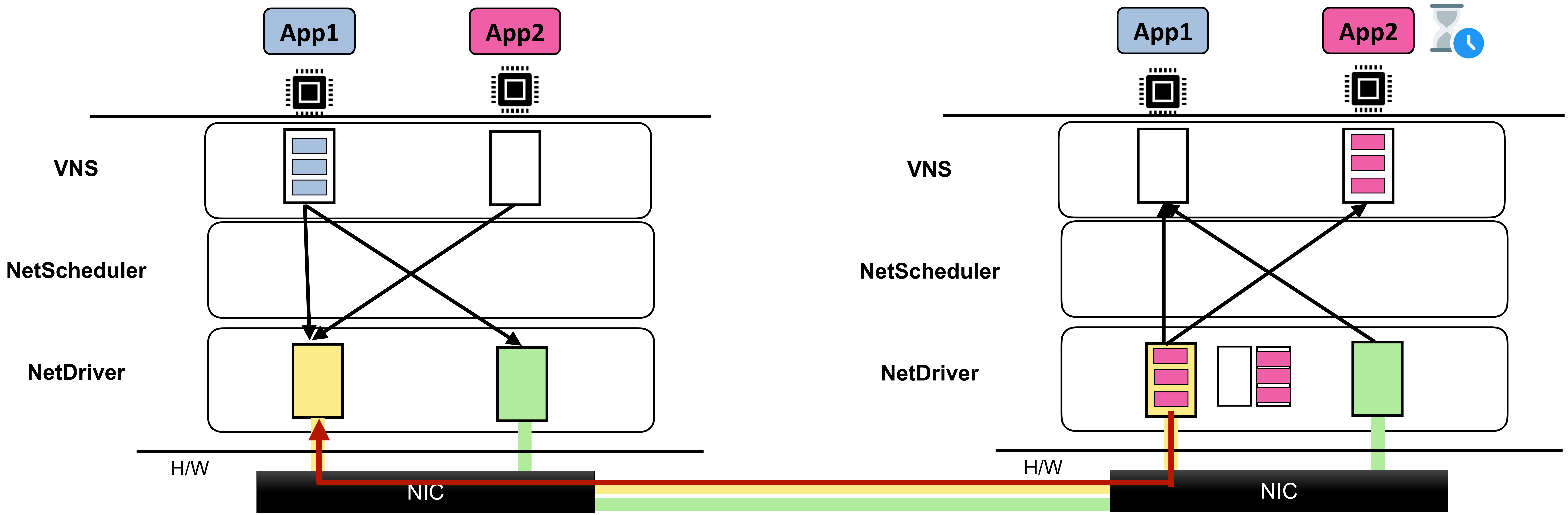
Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



Per-virtual socket response queues associated with each channel

NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

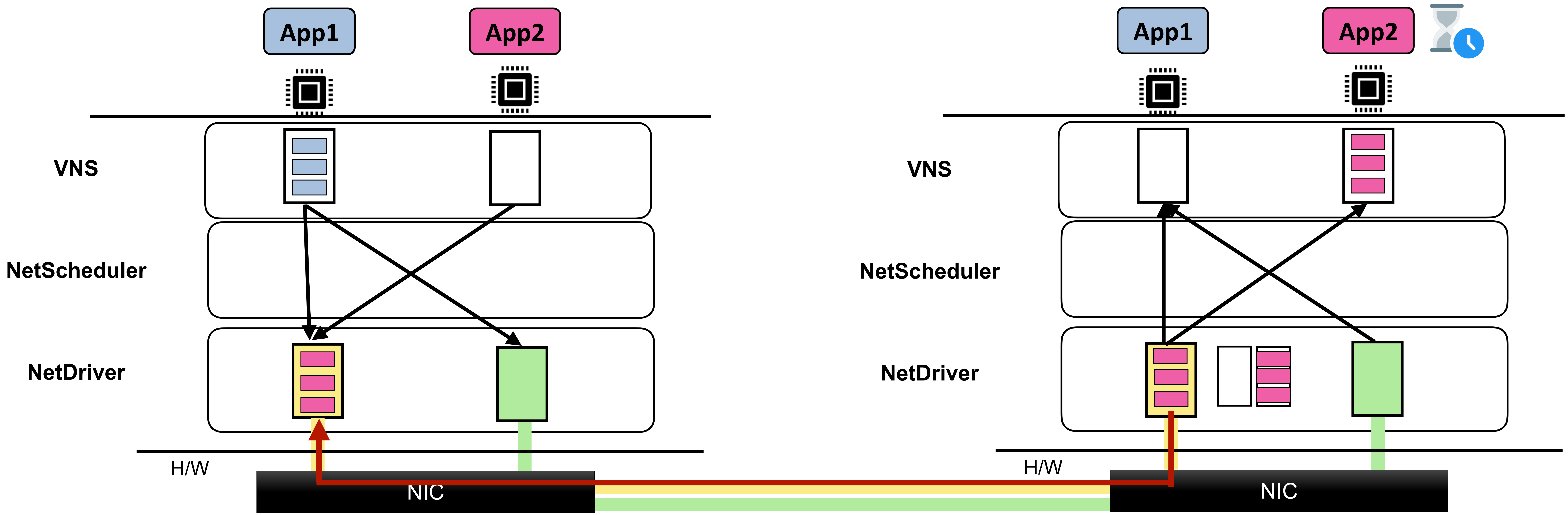


Per-virtual socket response queues associated with each channel

Piggybacks on transport-level flow control to apply back pressure

NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

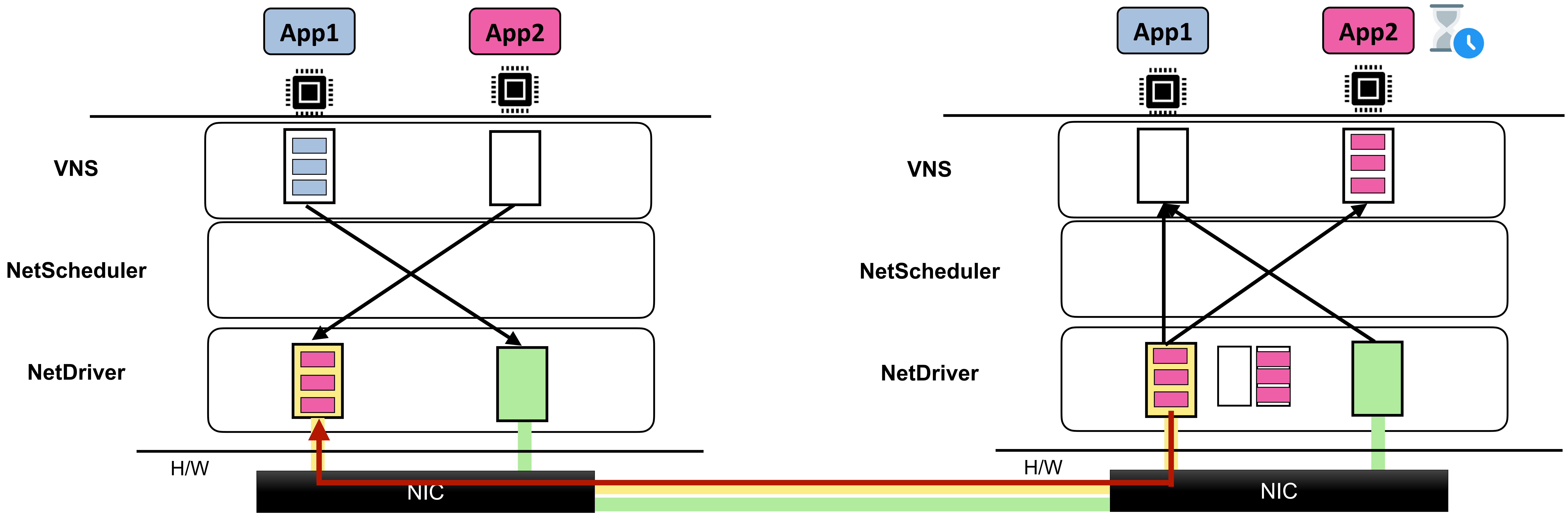


Per-virtual socket response queues associated with each channel

Piggybacks on transport-level flow control to apply back pressure

NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

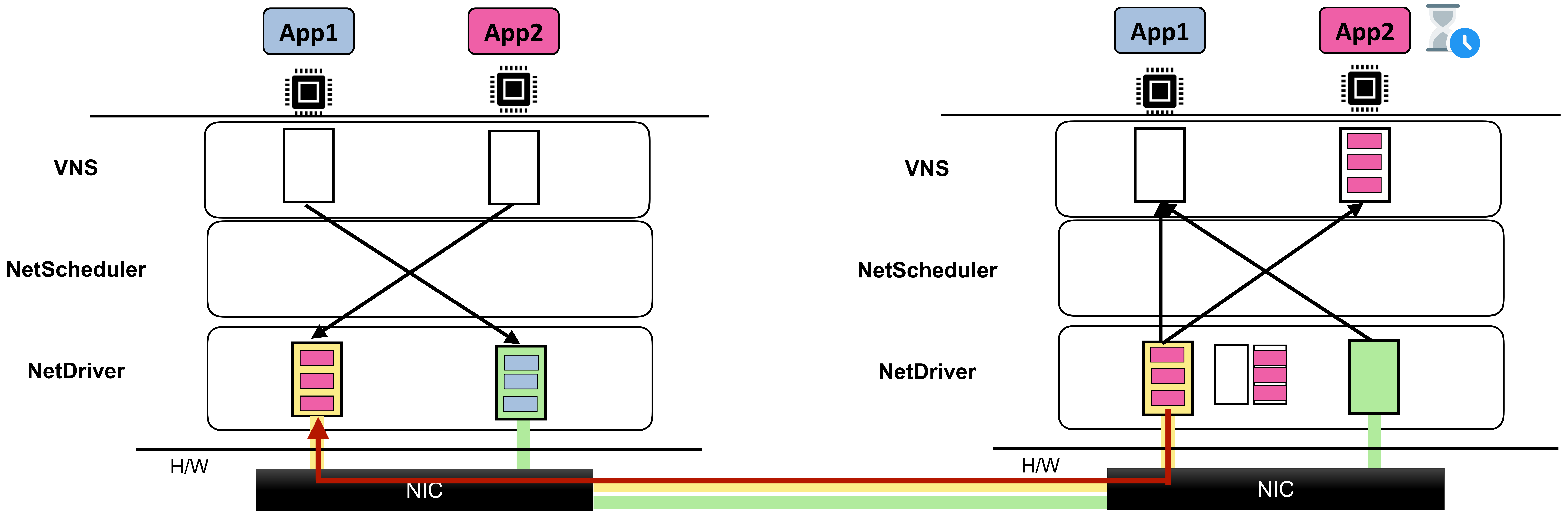


Per-virtual socket response queues associated with each channel

Piggybacks on transport-level flow control to apply back pressure

NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel

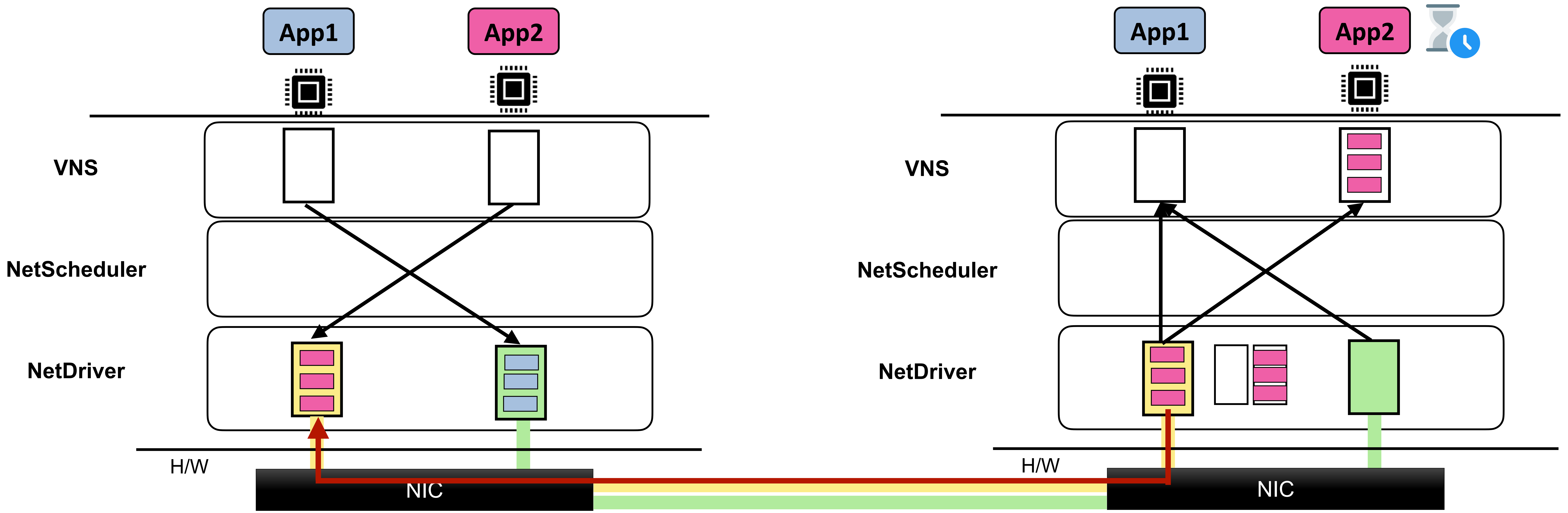


Per-virtual socket response queues associated with each channel

Piggybacks on transport-level flow control to apply back pressure

NetChannel End-to-End Operation

Challenge: Avoiding Head-of-Line (HoL) blocking when virtual interfaces share the same channel



Per-virtual socket response queues associated with each channel

Piggybacks on transport-level flow control to apply back pressure

For persistent HoL blocking: Virtual interface level flow control mechanism (see paper for details)

This Work

Limitations of Dedicated, Tightly Integrated, Static pipelines
Preclude network stacks from exploiting capabilities of modern hardware

NetChannel: New Architecture for Host Network Stacks
Disaggregates packet processing pipeline

Prototype NetChannel Implementation in the Linux Network Stack
Demonstrate new operating points through experimental evaluation

Evaluation Setup

- **Implemented as a kernel module in Linux**
- **To push the bottleneck to the Host Network Stack**
  - Two 32-core servers directly connected 200Gbps (8 cores per NUMA node)
  - Minimal application-level processing
- **Demonstrate NetChannel achieves new operating points**
  - Saturate a 200Gbps using a single socket
  - Scale short message processing throughput linearly
  - Achieve µs-scale tail latency for L-apps when colocated with T-apps
- **Evaluation on real-world applications**
  - Redis: in-memory database
  - SPDK: remote storage stacks

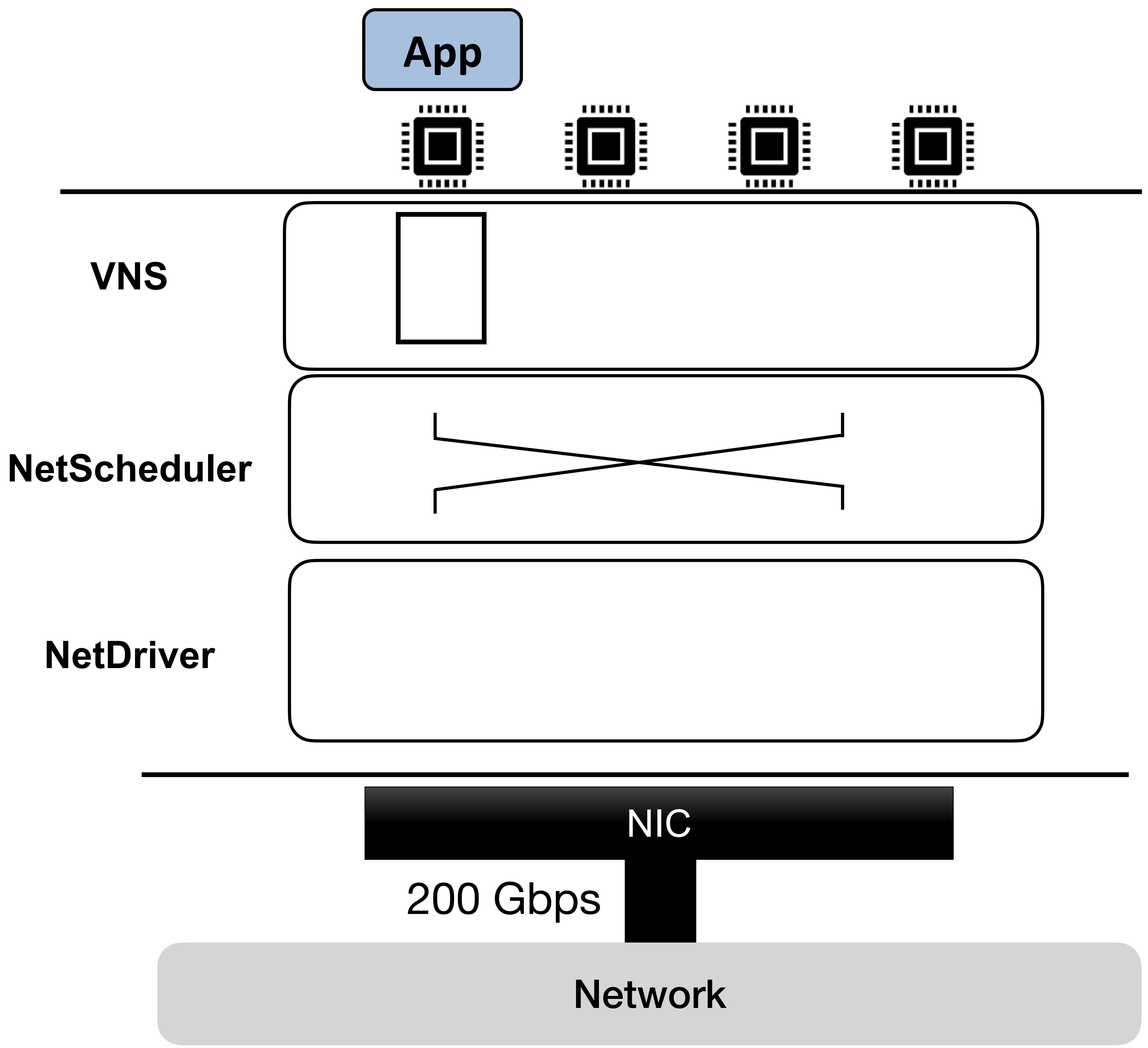
Evaluation Setup

- Implemented as a kernel module in Linux
- To push the bottleneck to the Host Network Stack
  - Two 32-core servers directly connected 200Gbps (8 cores per NUMA node)
  - Minimal application-level processing

See Paper for more detailed Evaluation

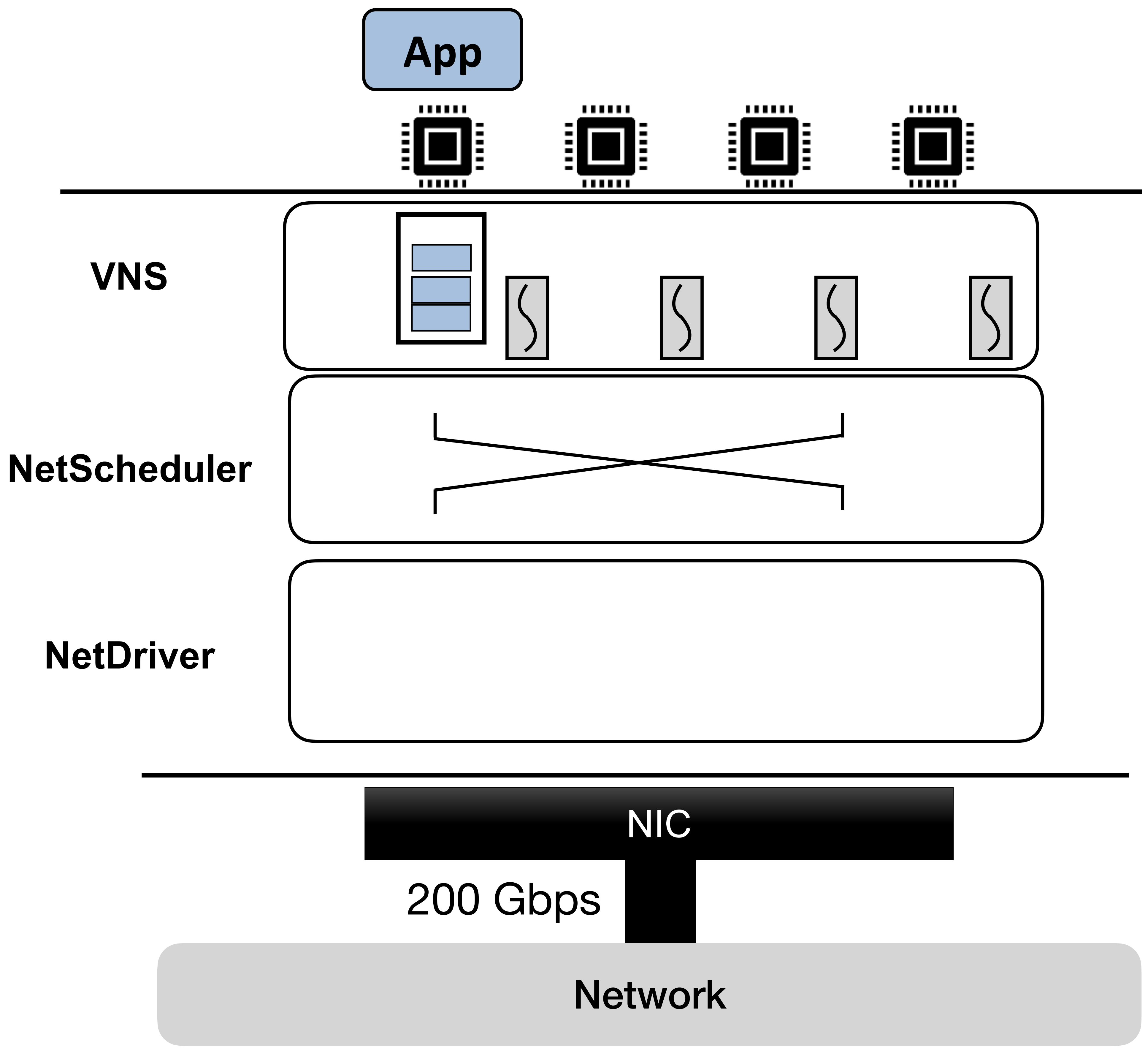
- **Demonstrate NetChannel achieves new operating points**
  - Saturate a 200Gbps using a single socket
  - Scale short message processing throughput linearly
  - Achieve µs-scale tail latency for L-apps when colocated with T-apps
- Evaluation on real-world applications
  - Redis: in-memory database
  - SPDK: remote storage stacks

NetChannel: Towards Terabit Ethernet



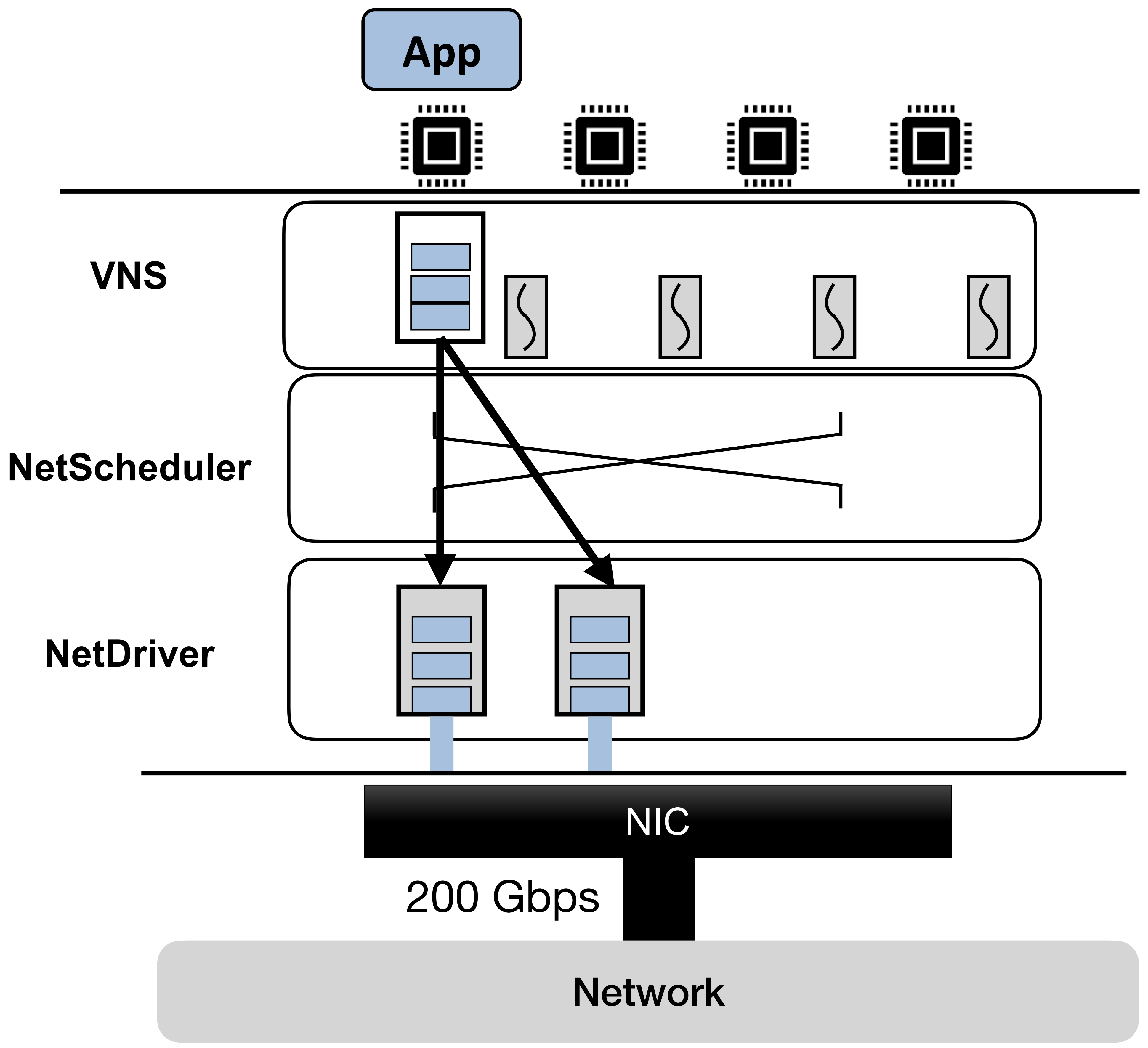
Long flow

NetChannel: Towards Terabit Ethernet



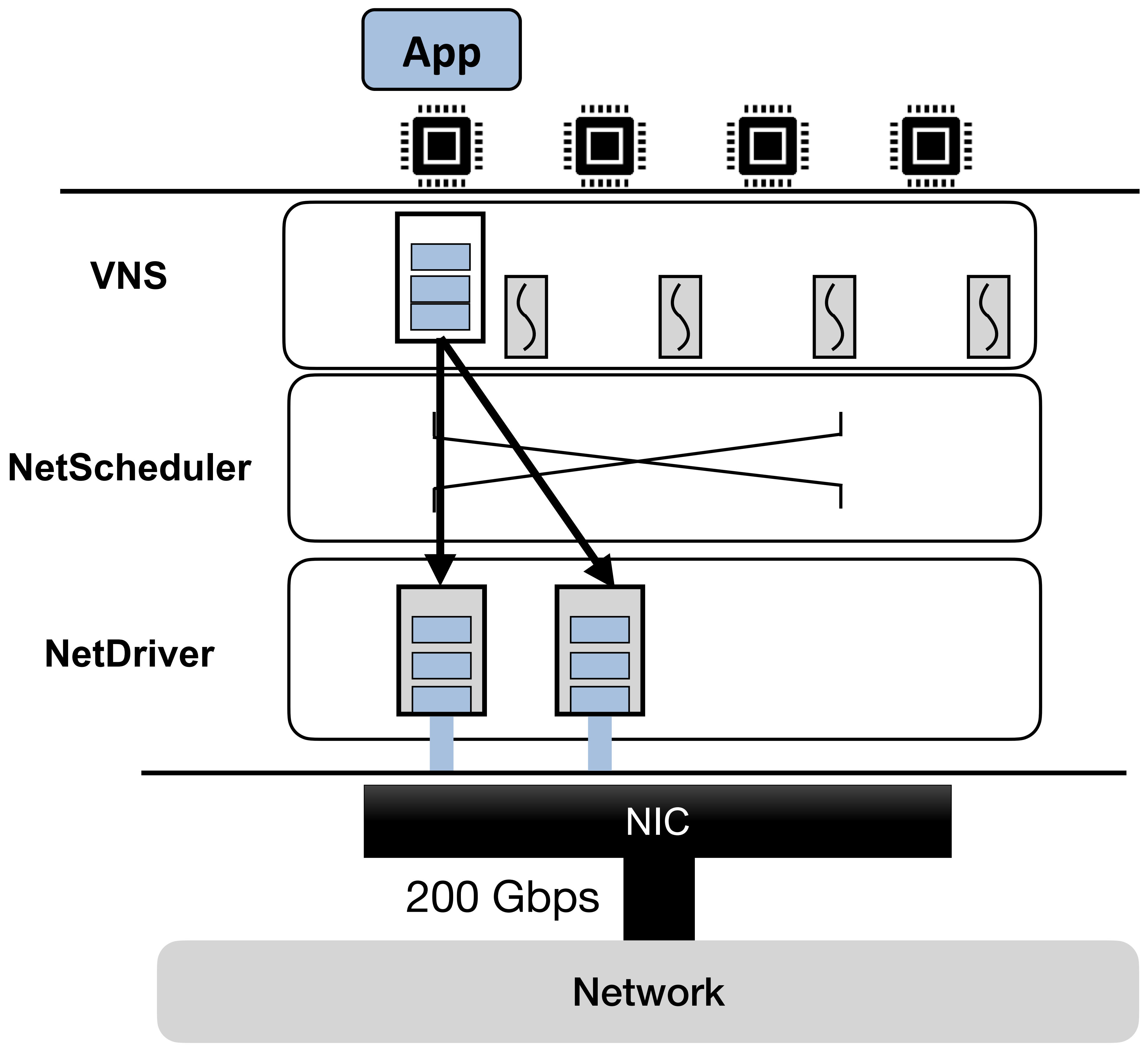
Long flow

NetChannel: Towards Terabit Ethernet

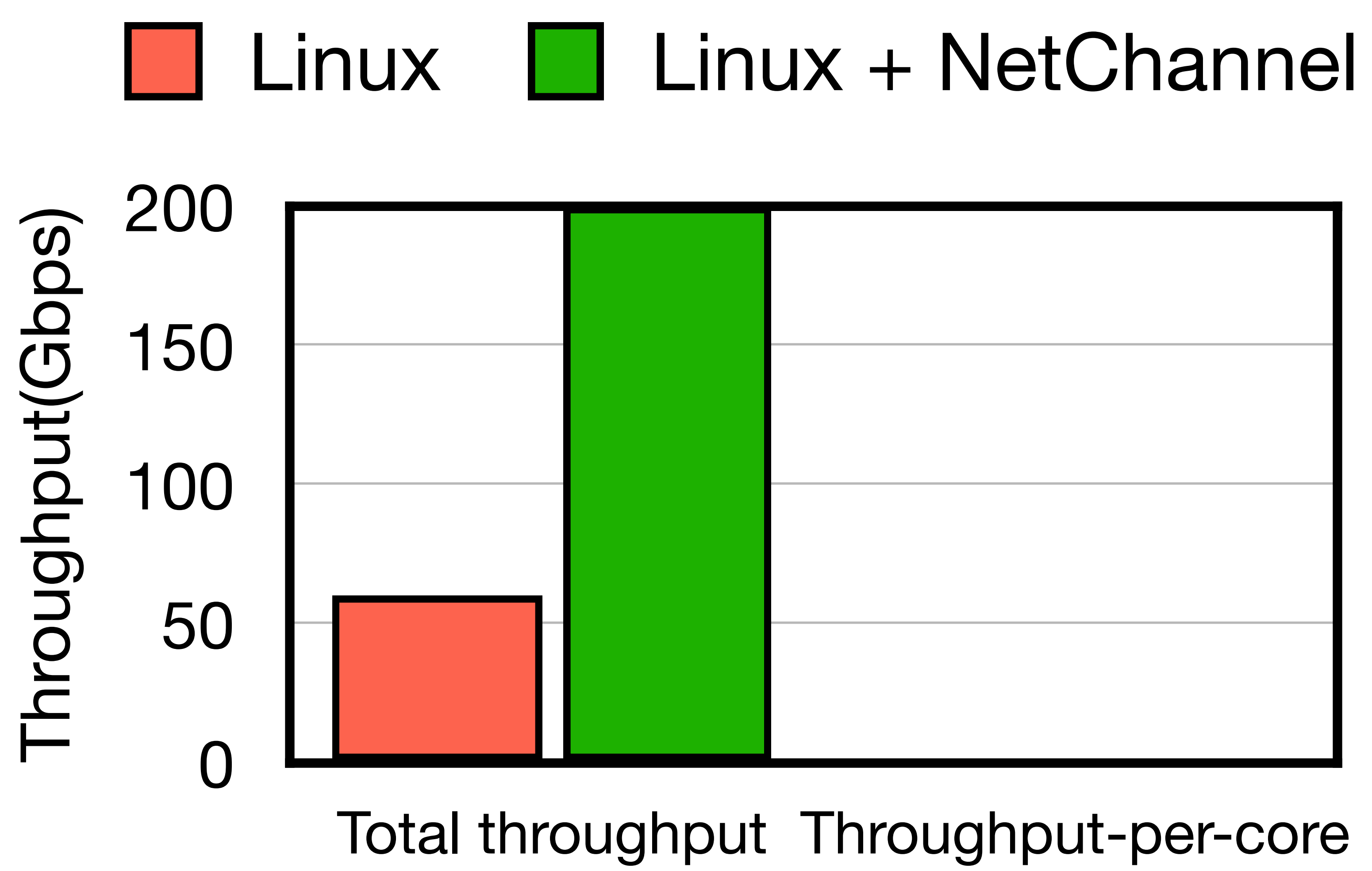


Long flow

NetChannel: Towards Terabit Ethernet

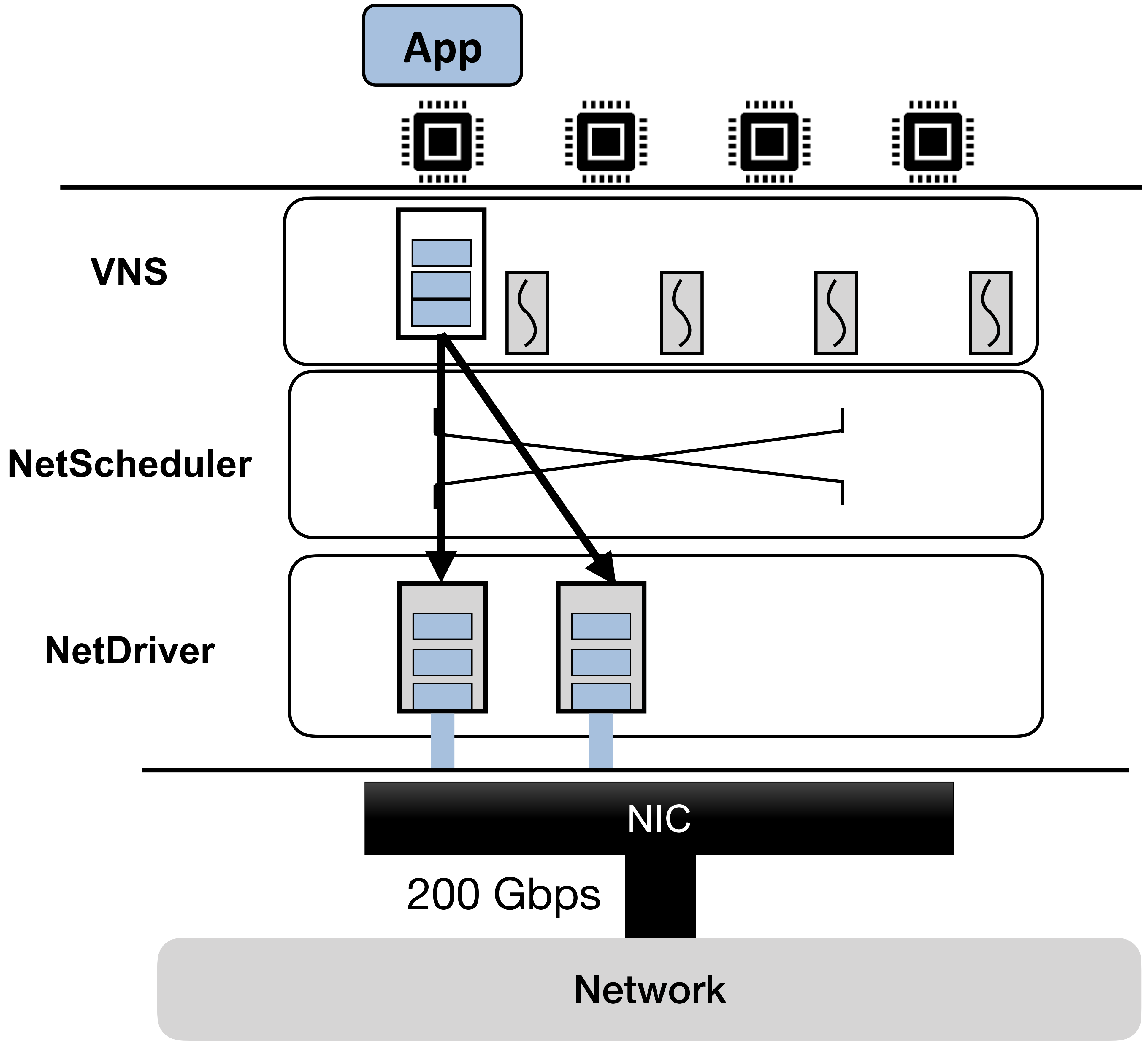


Long flow

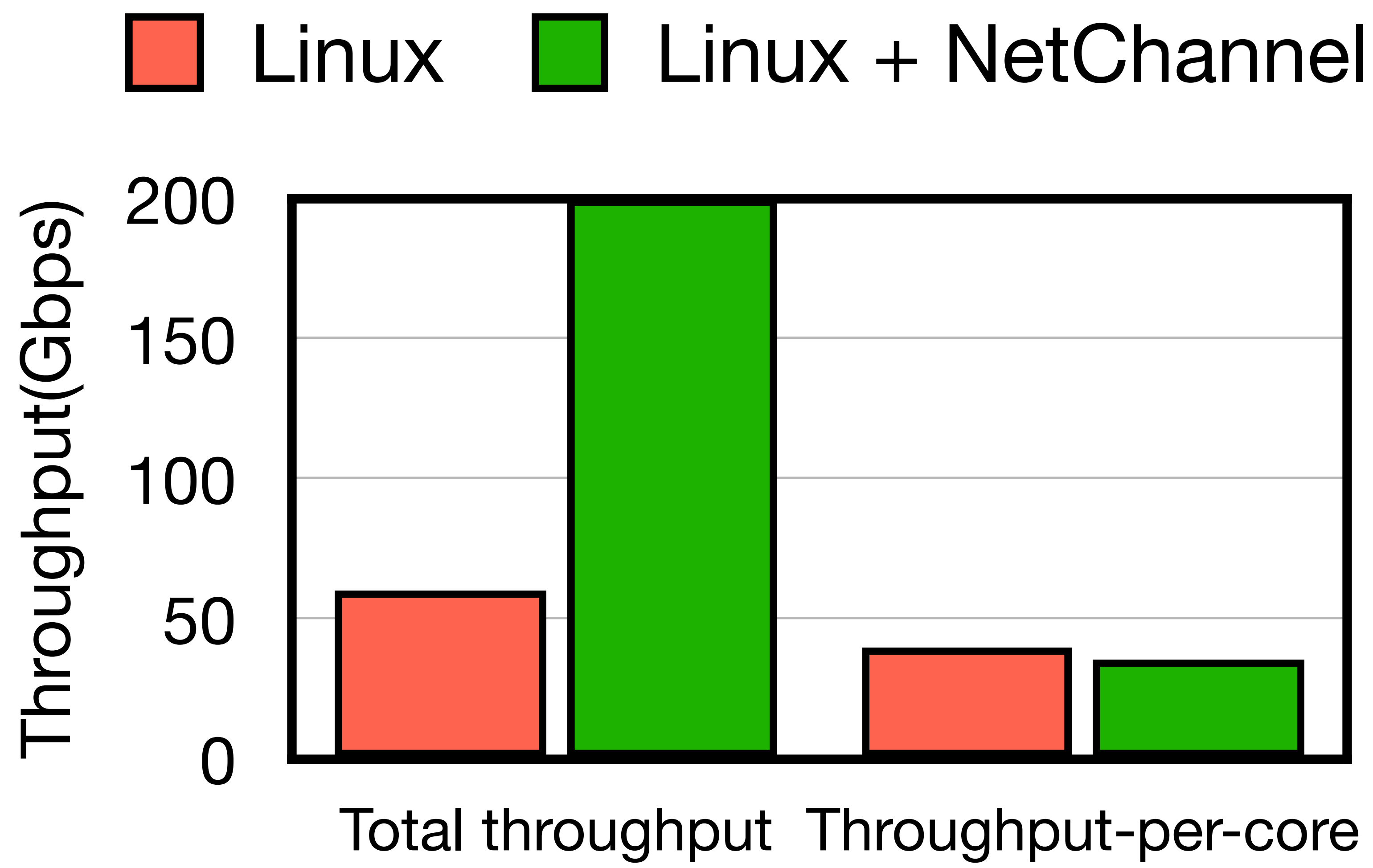


NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet

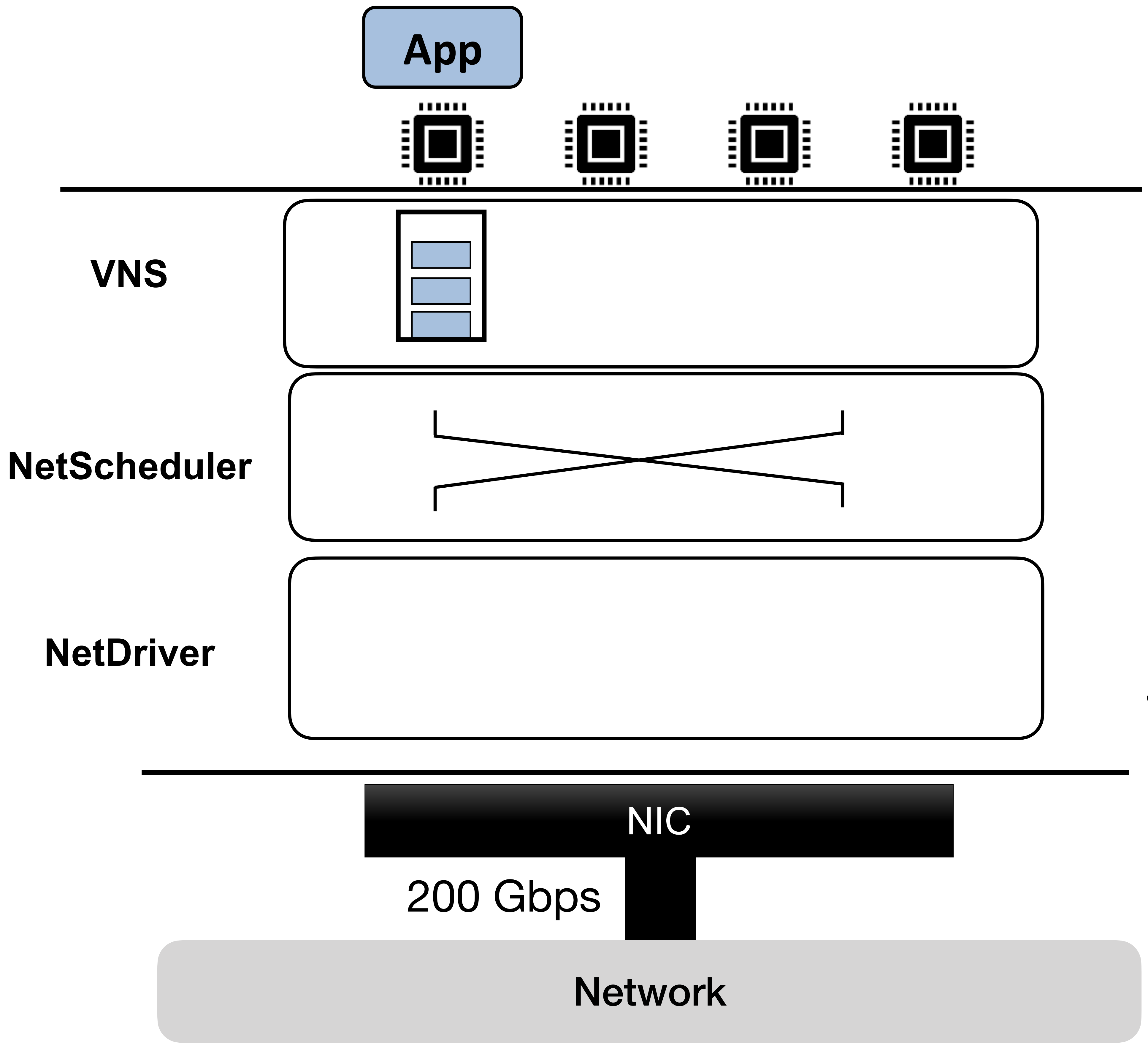


Long flow

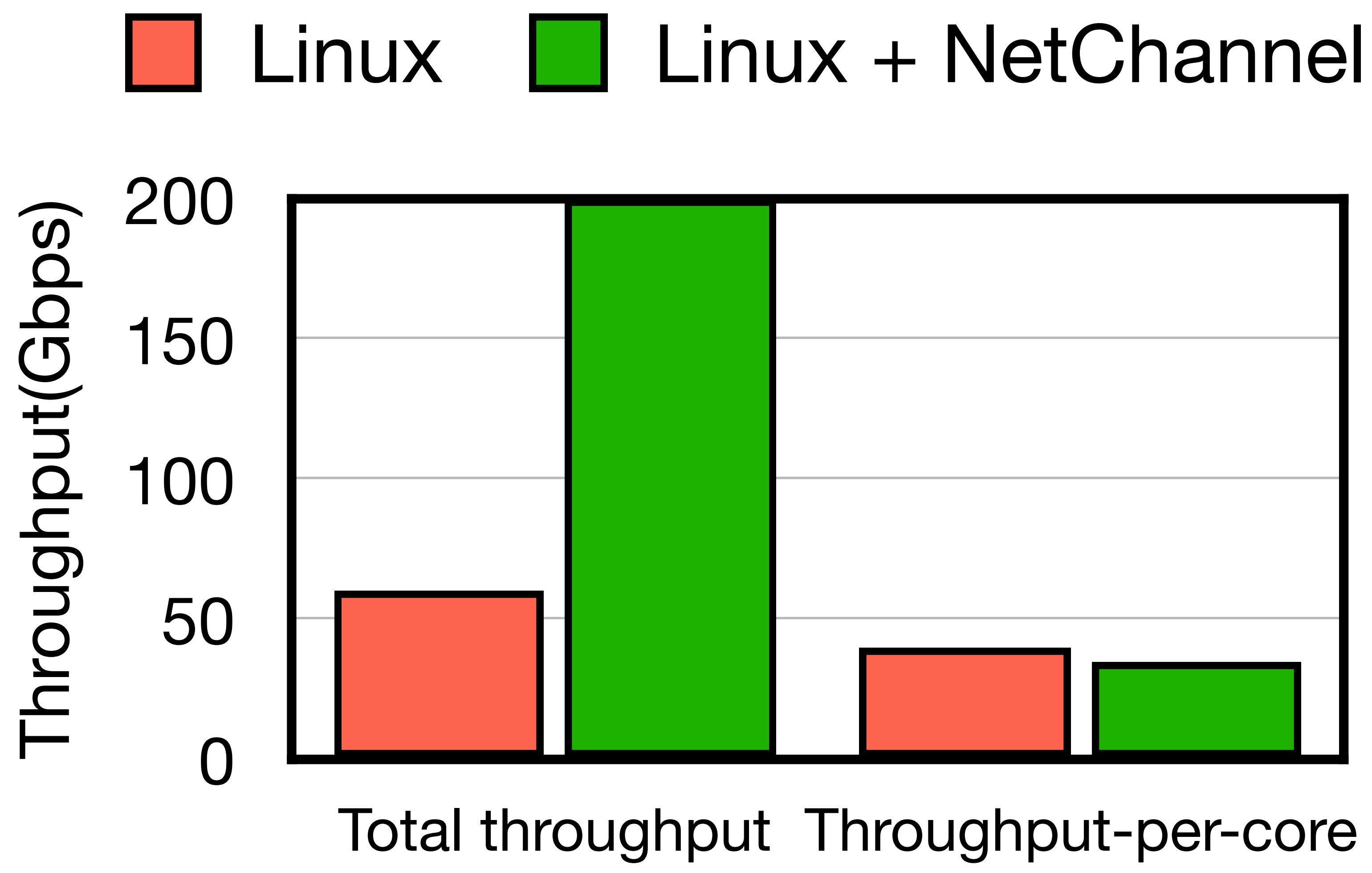


NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet



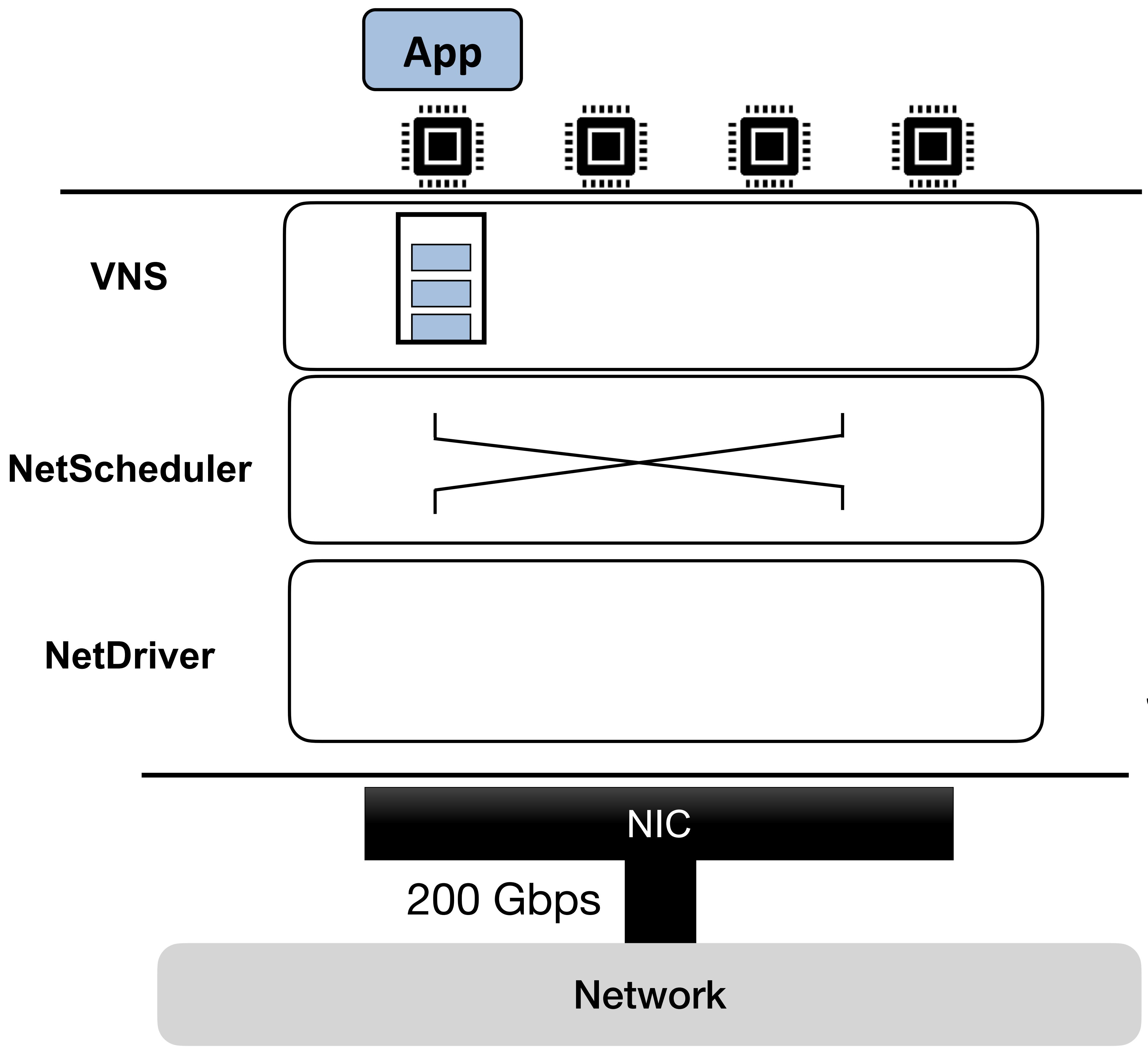
Long flow



Short Messages

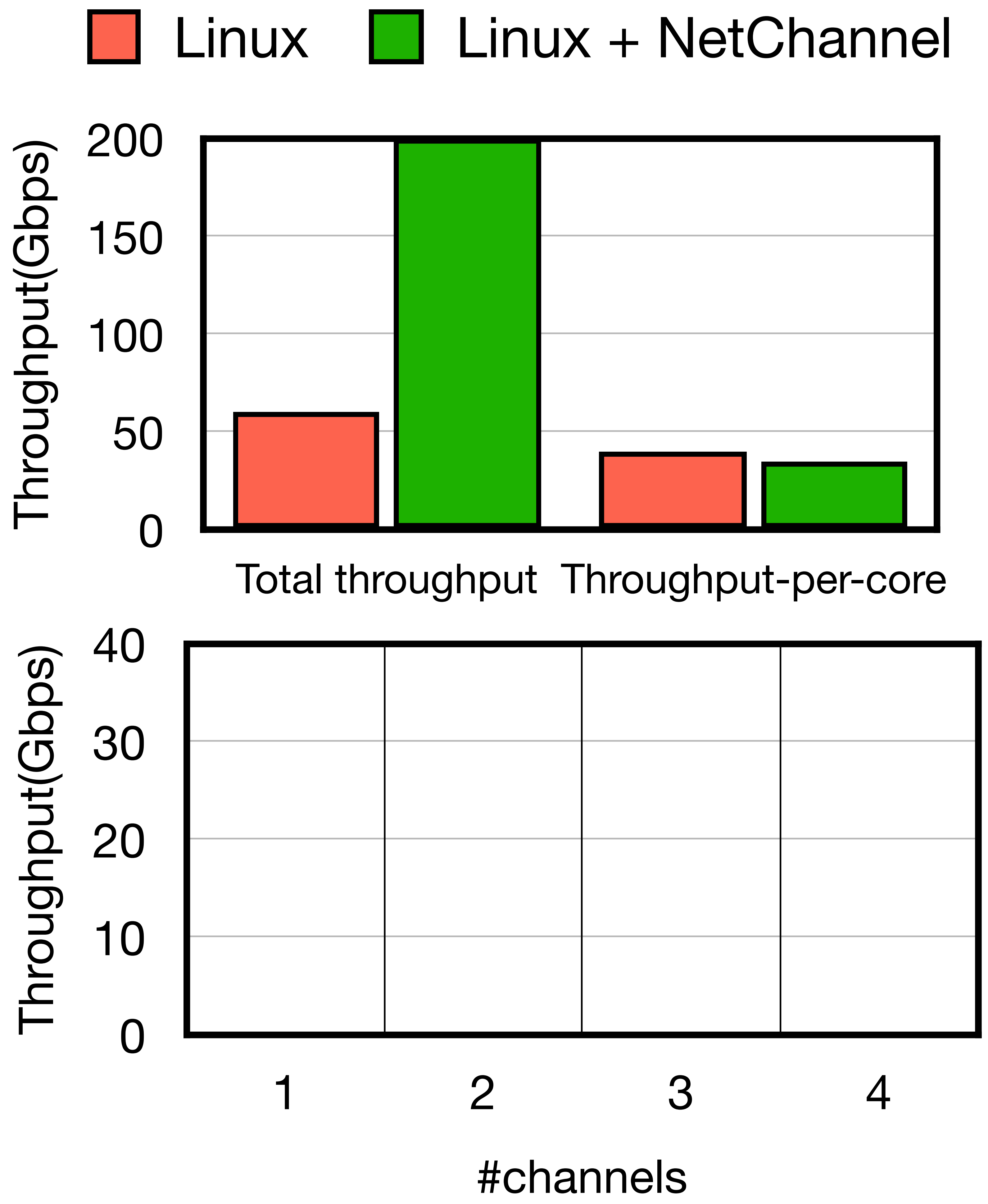
NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet



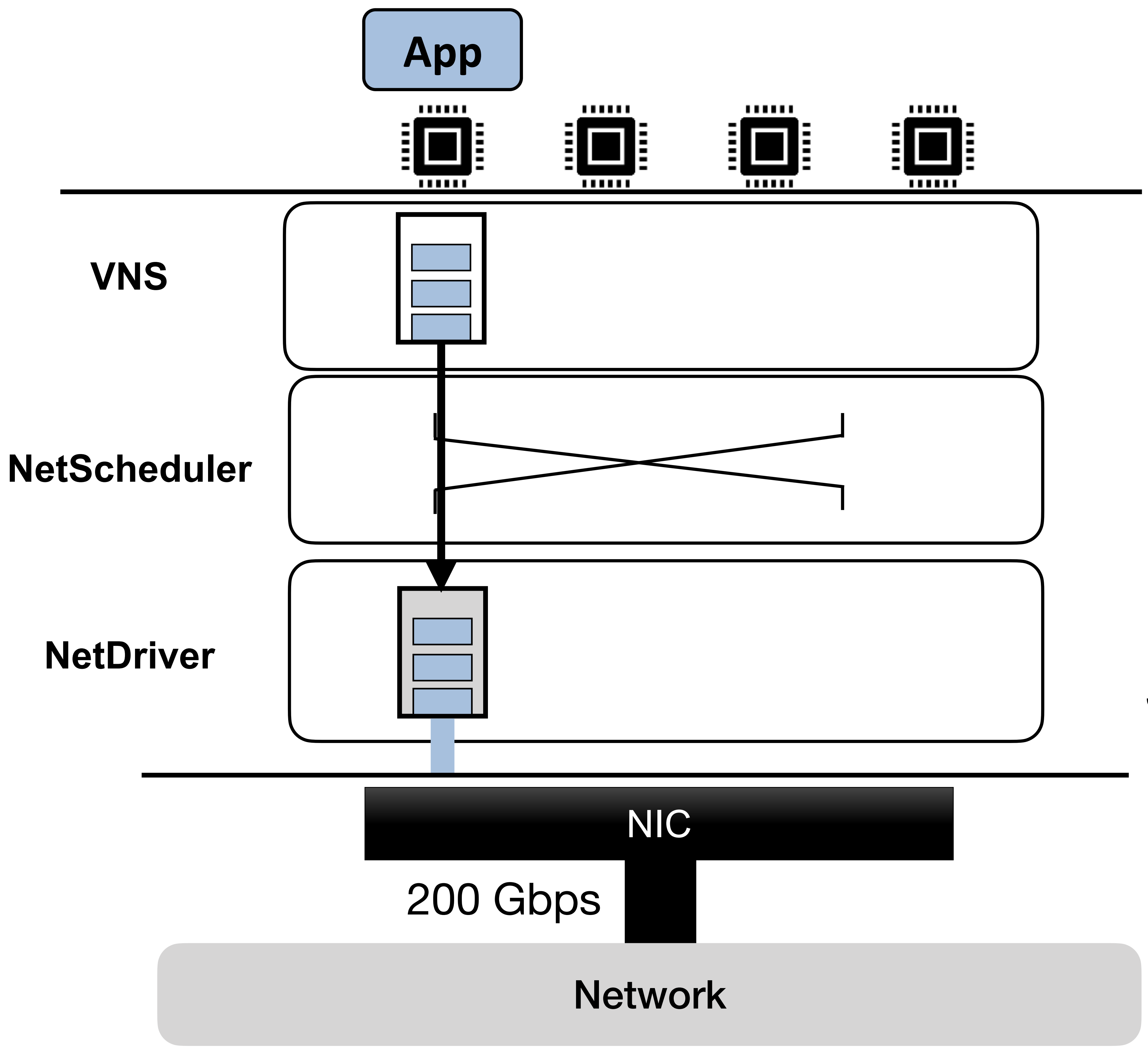
Long flow

Short Messages



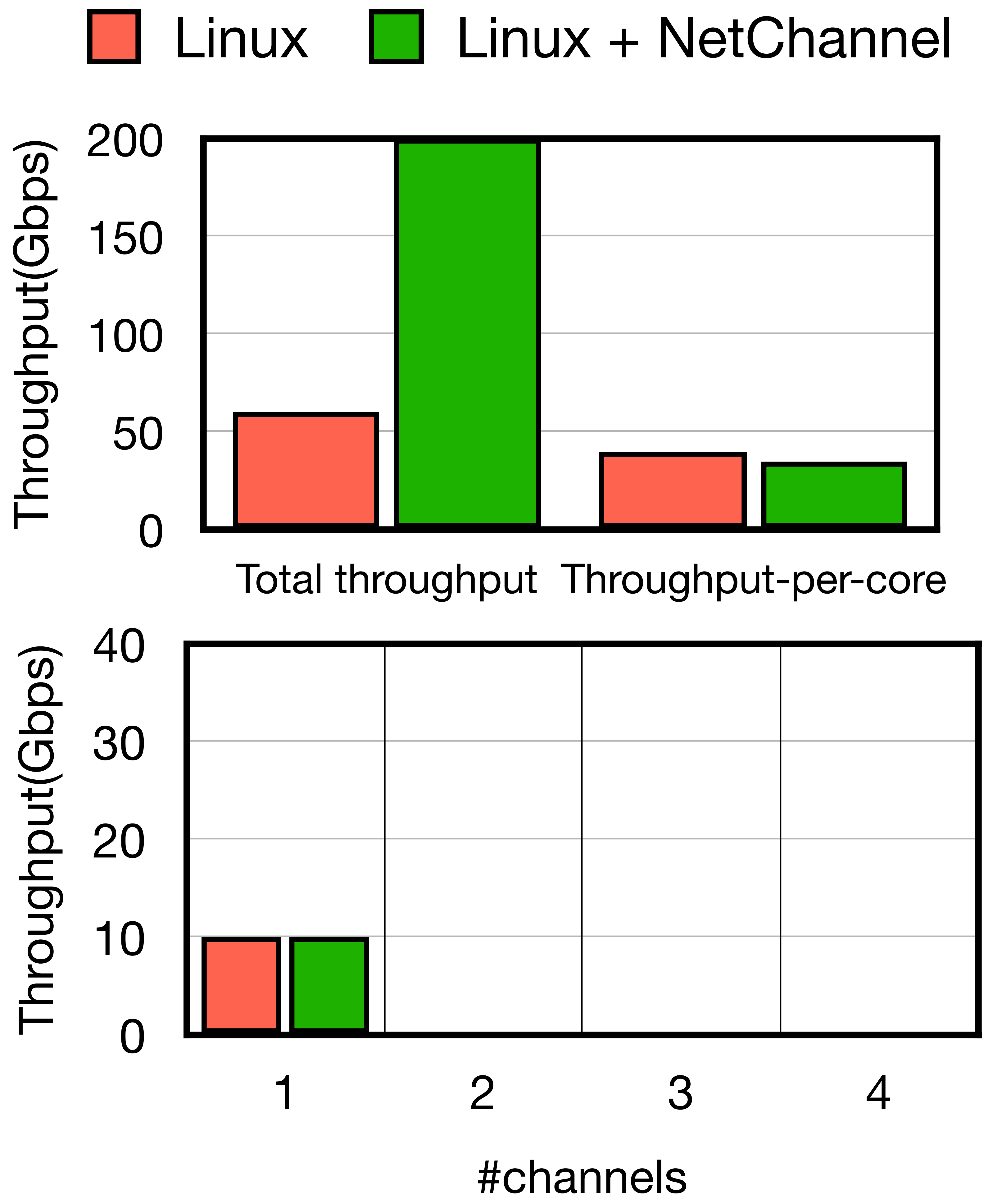
NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet



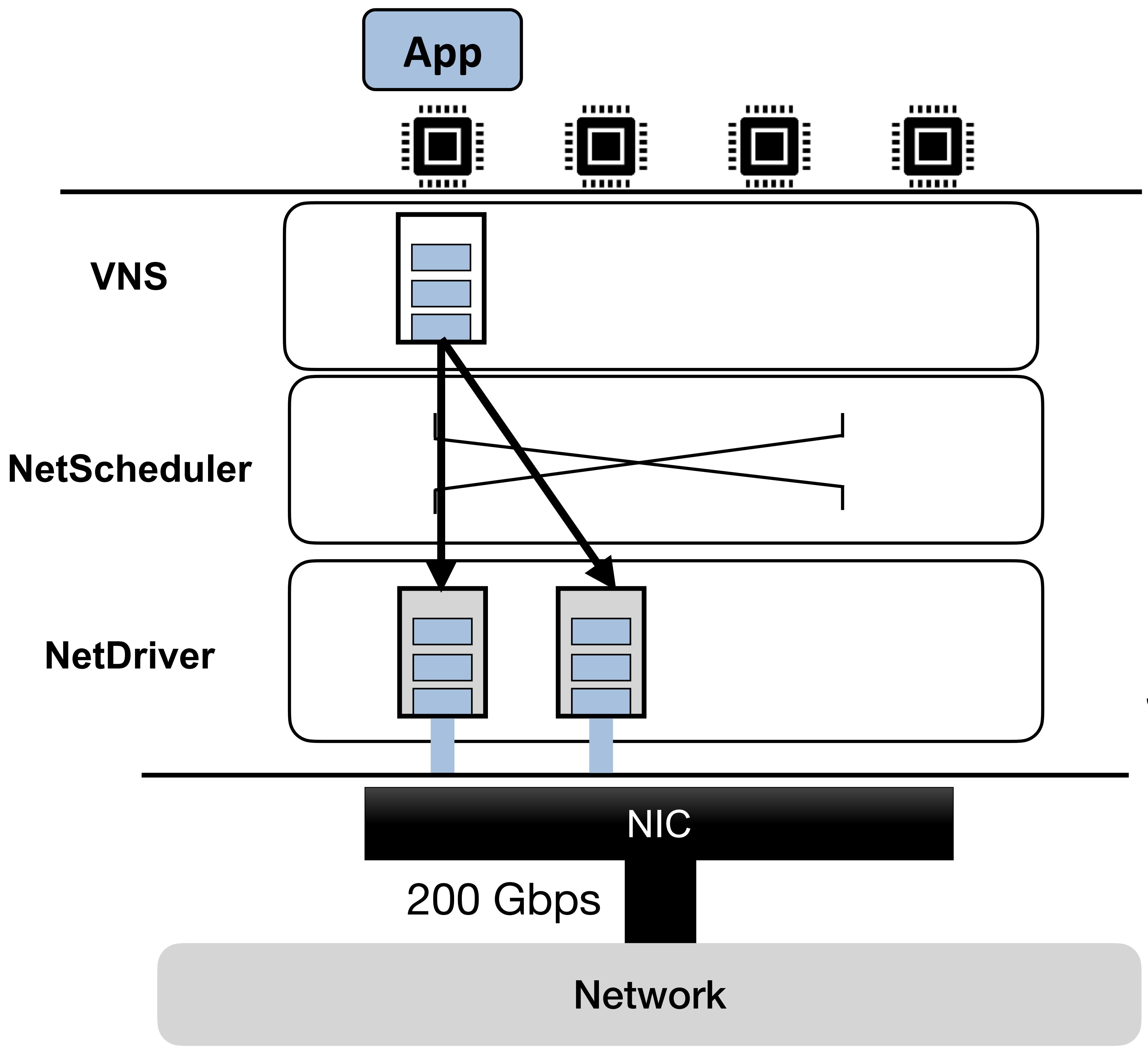
Long flow

Short Messages



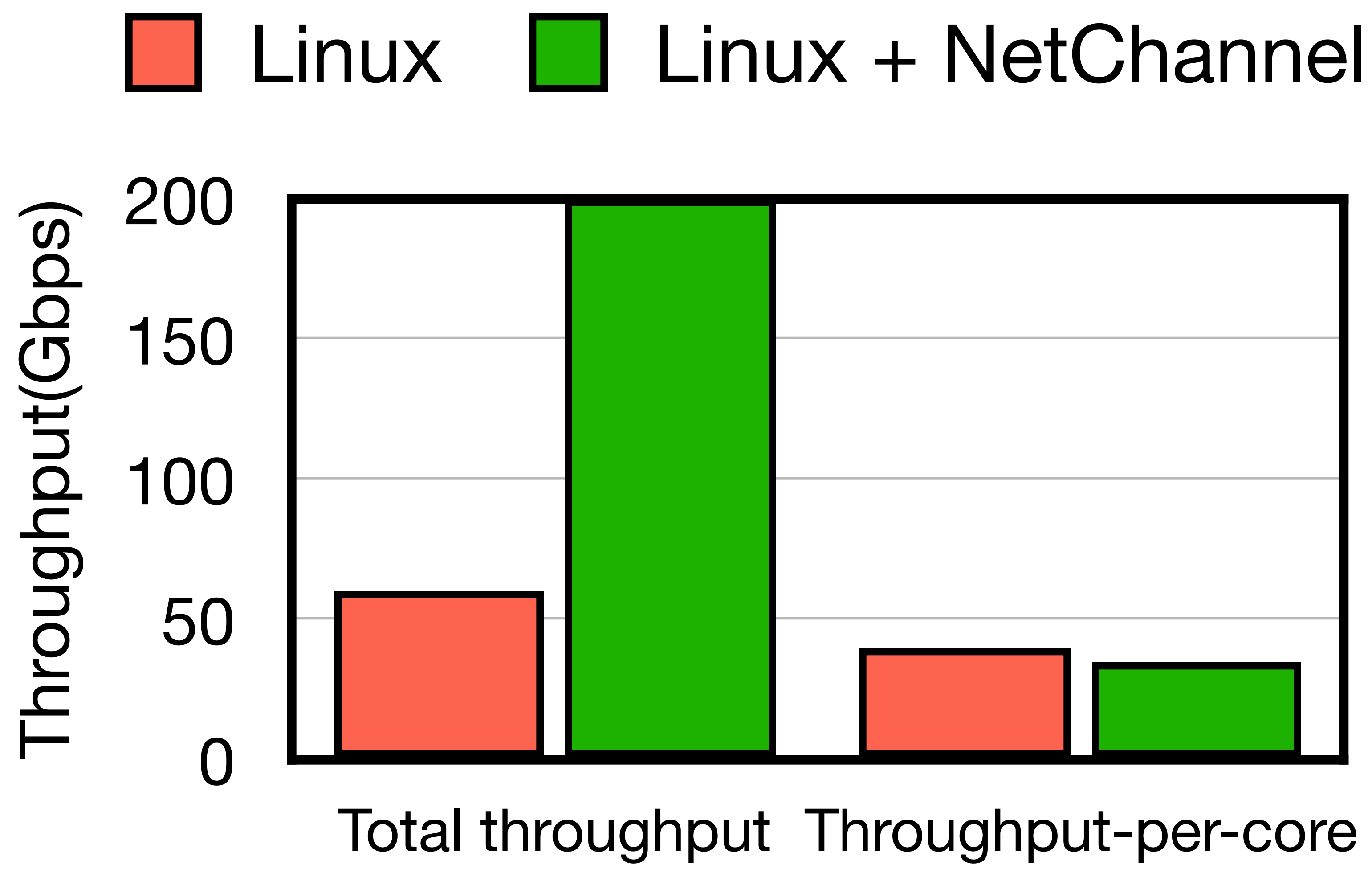
NetChannel enables saturating a Terabit Ethernet link with a single socket

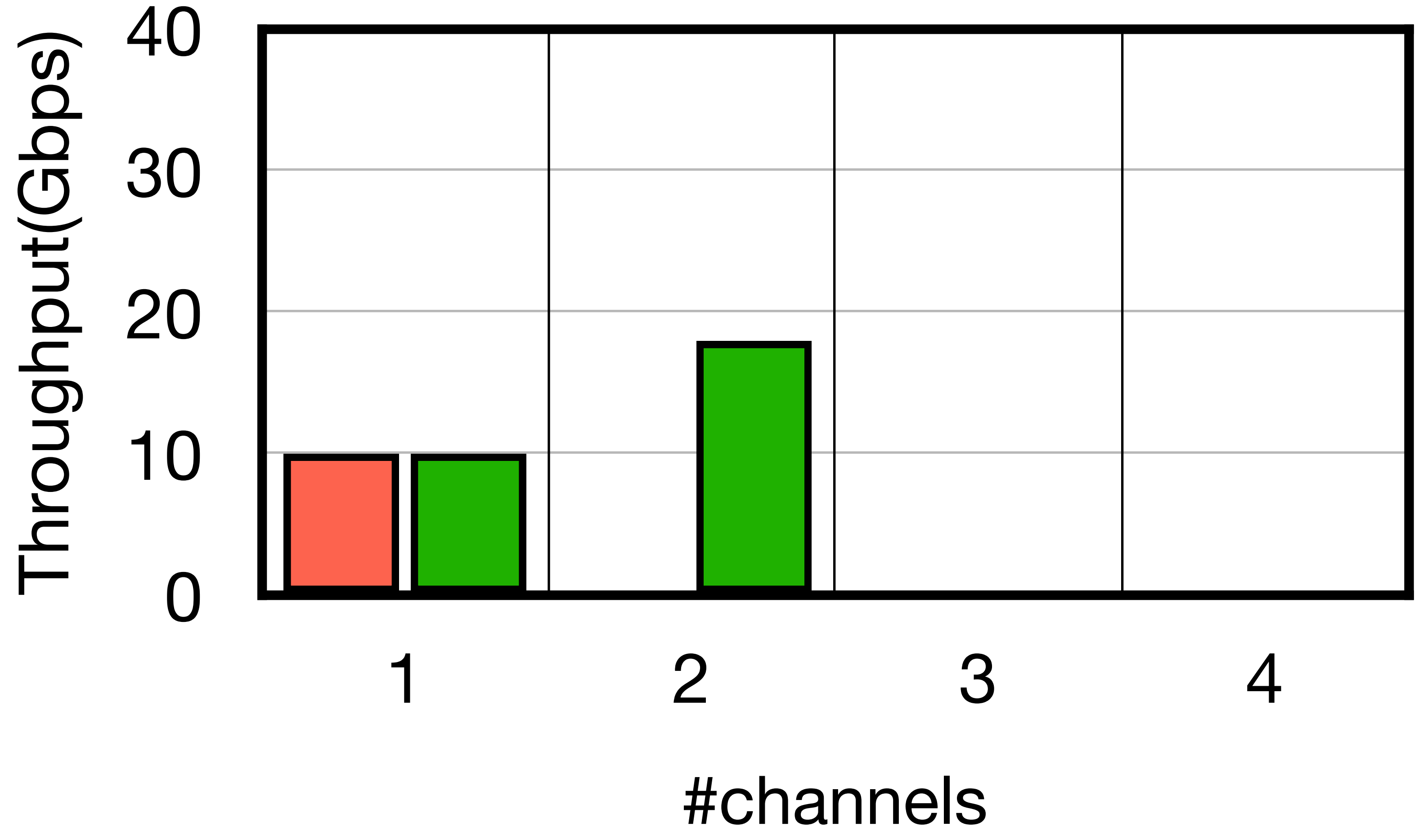
NetChannel: Towards Terabit Ethernet



Long flow

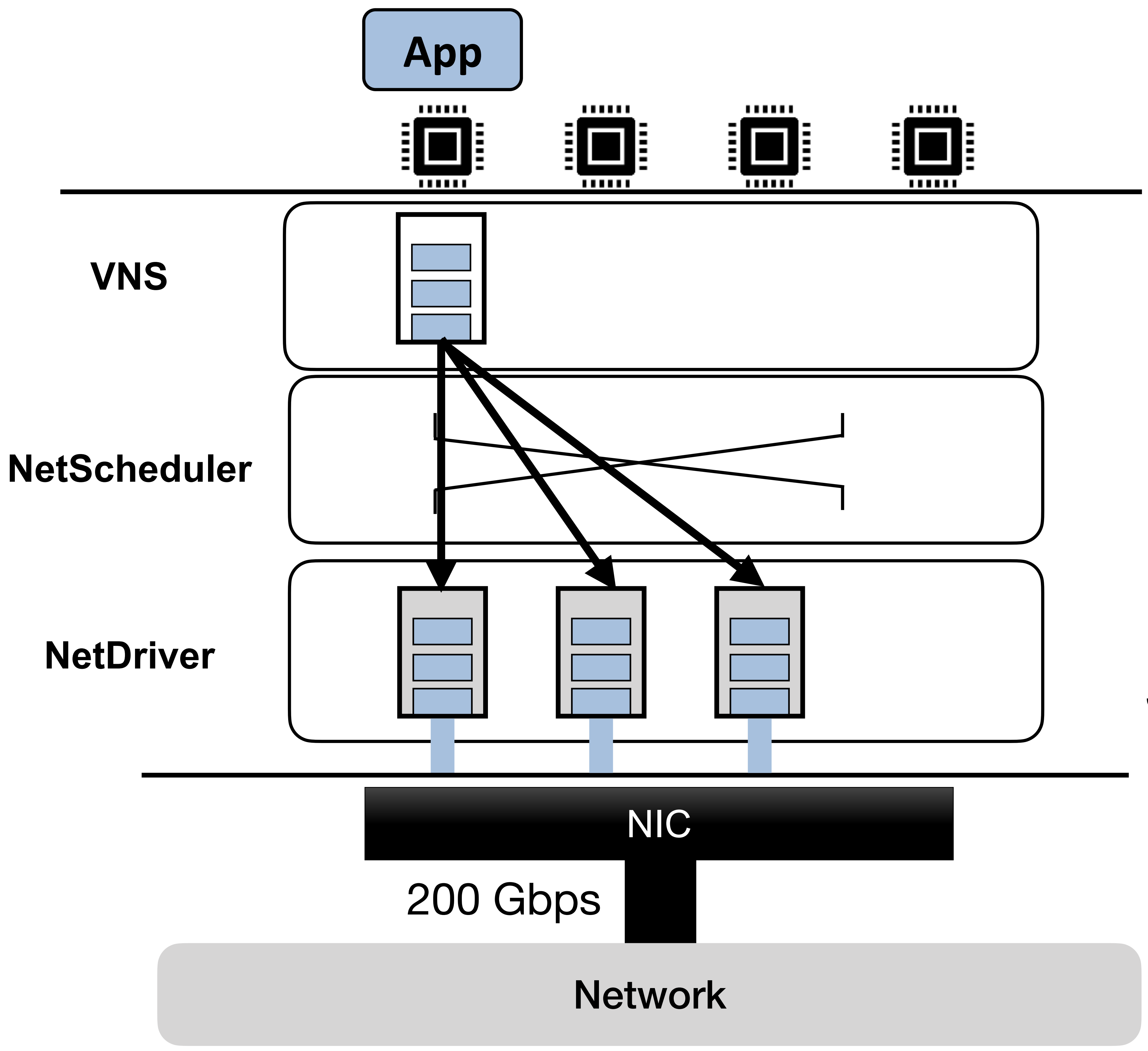
Short Messages





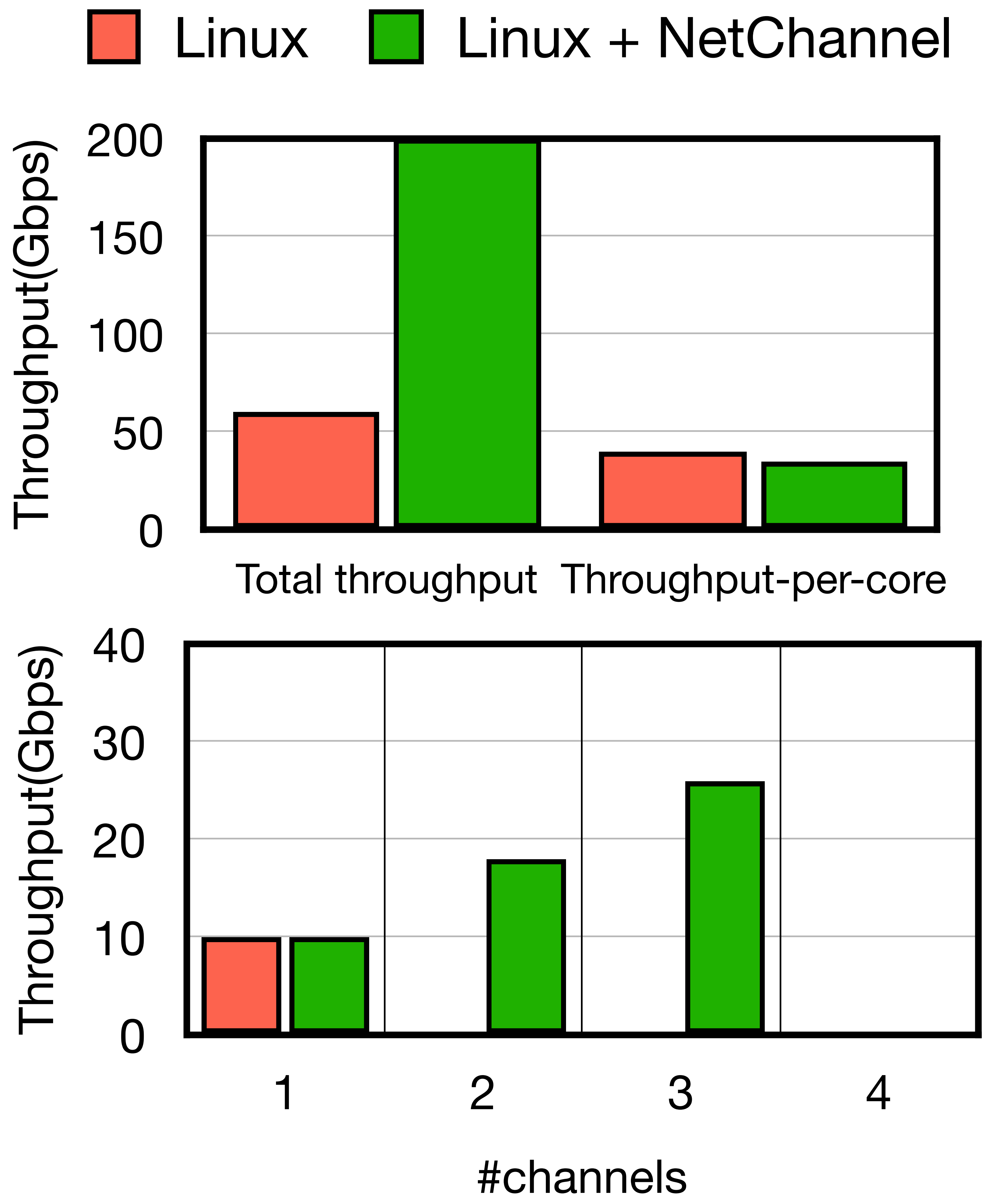
NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet



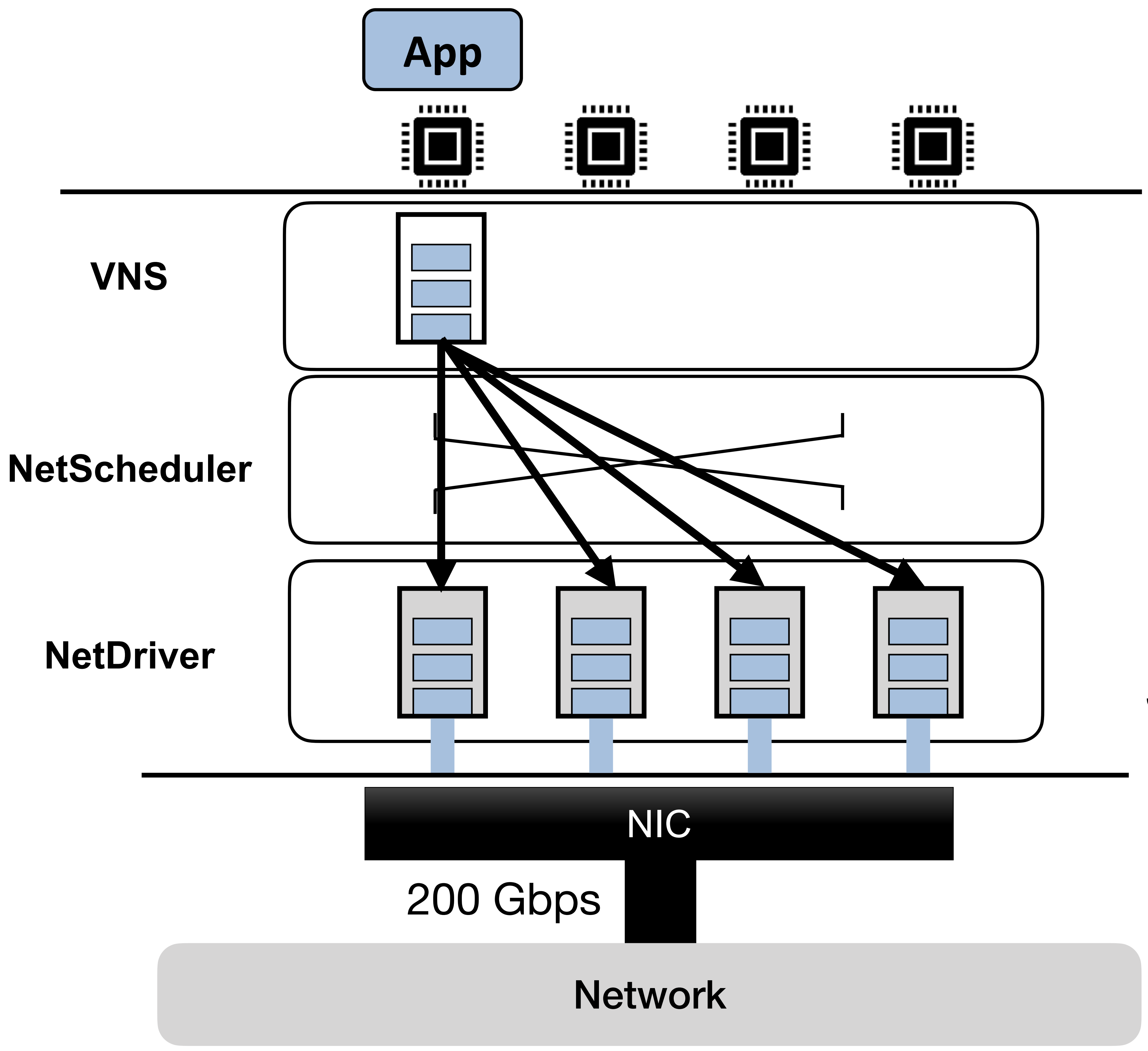
Long flow

Short Messages



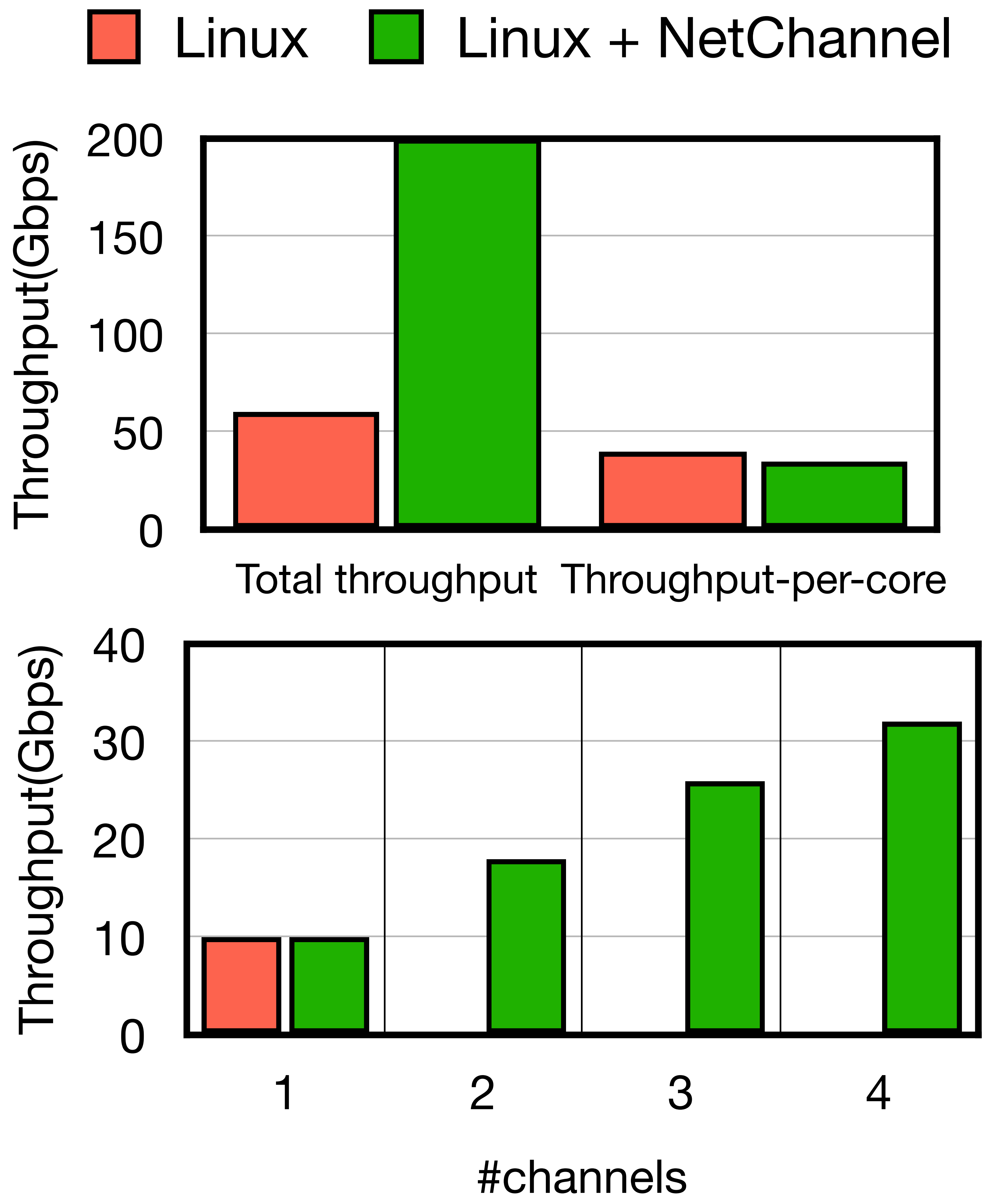
NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet



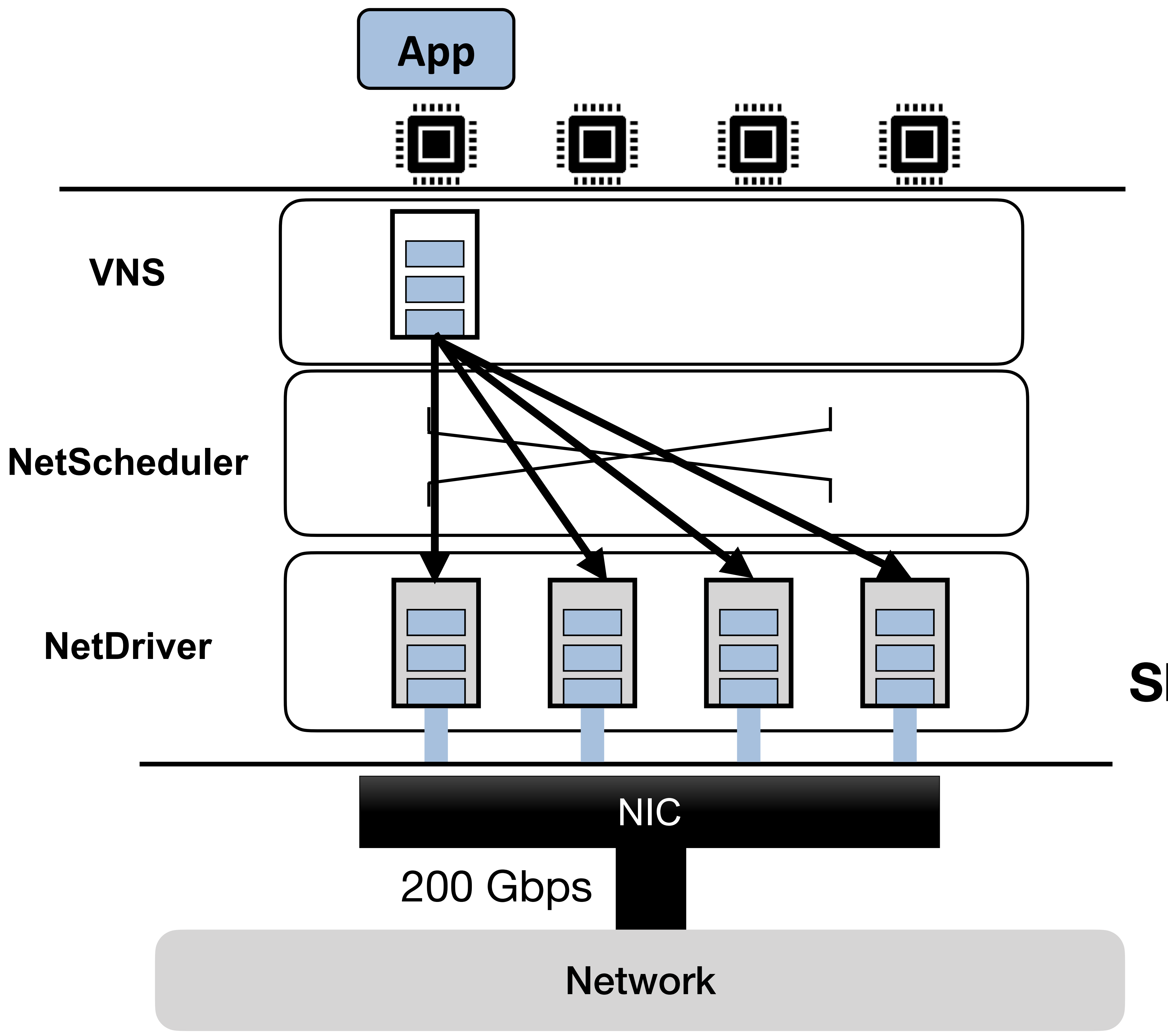
Long flow

Short Messages



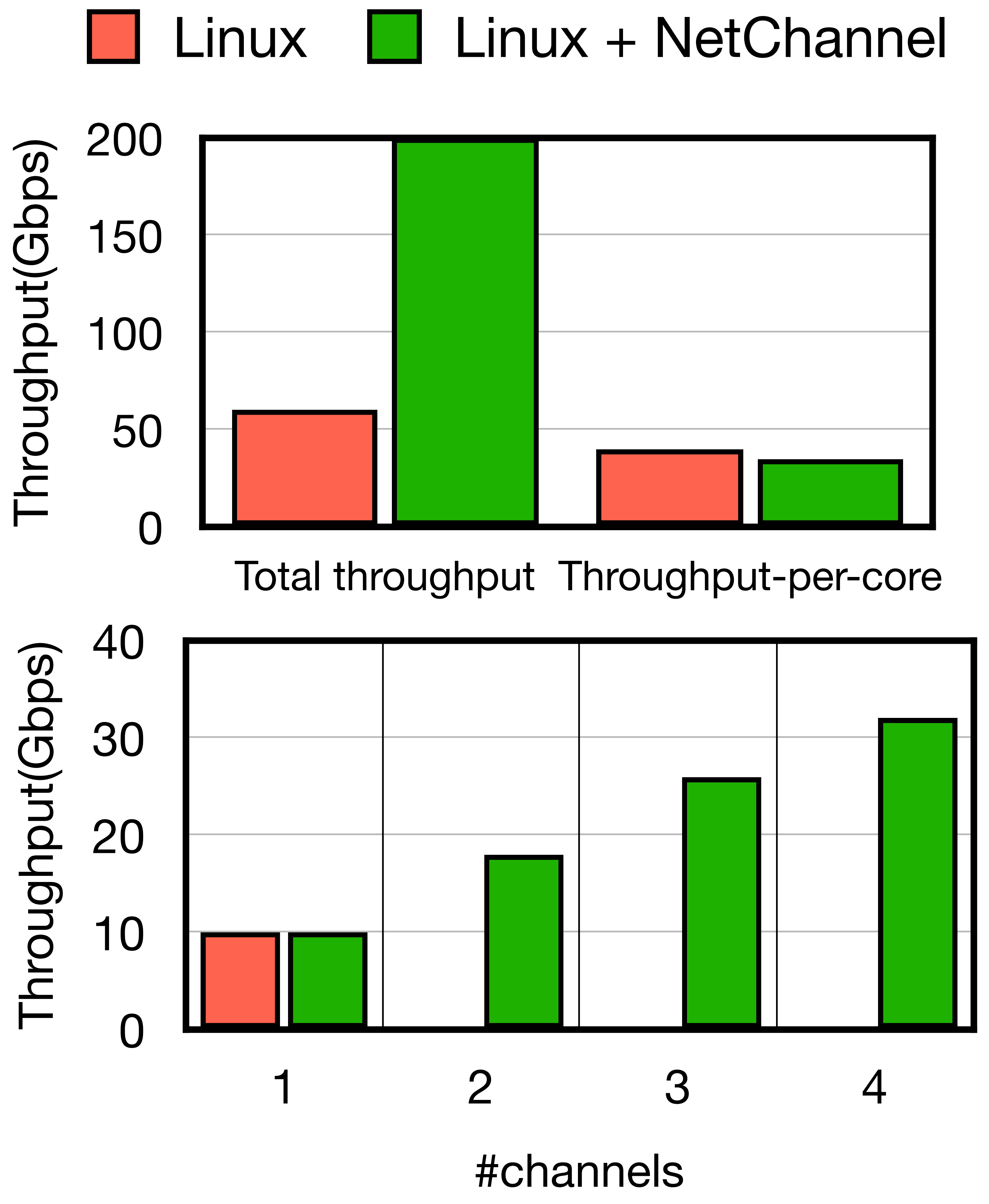
NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel: Towards Terabit Ethernet



Long flow

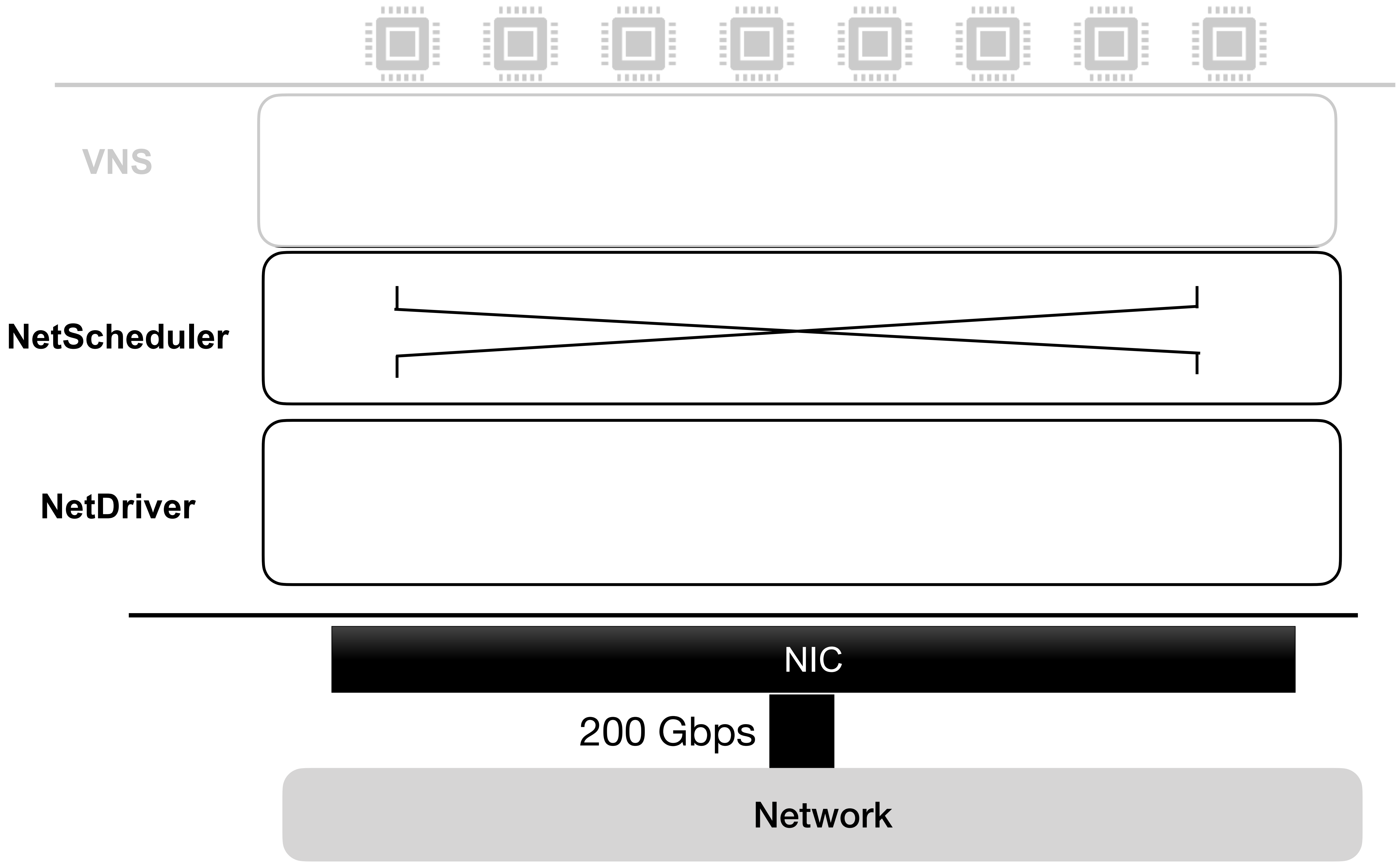
Short Messages



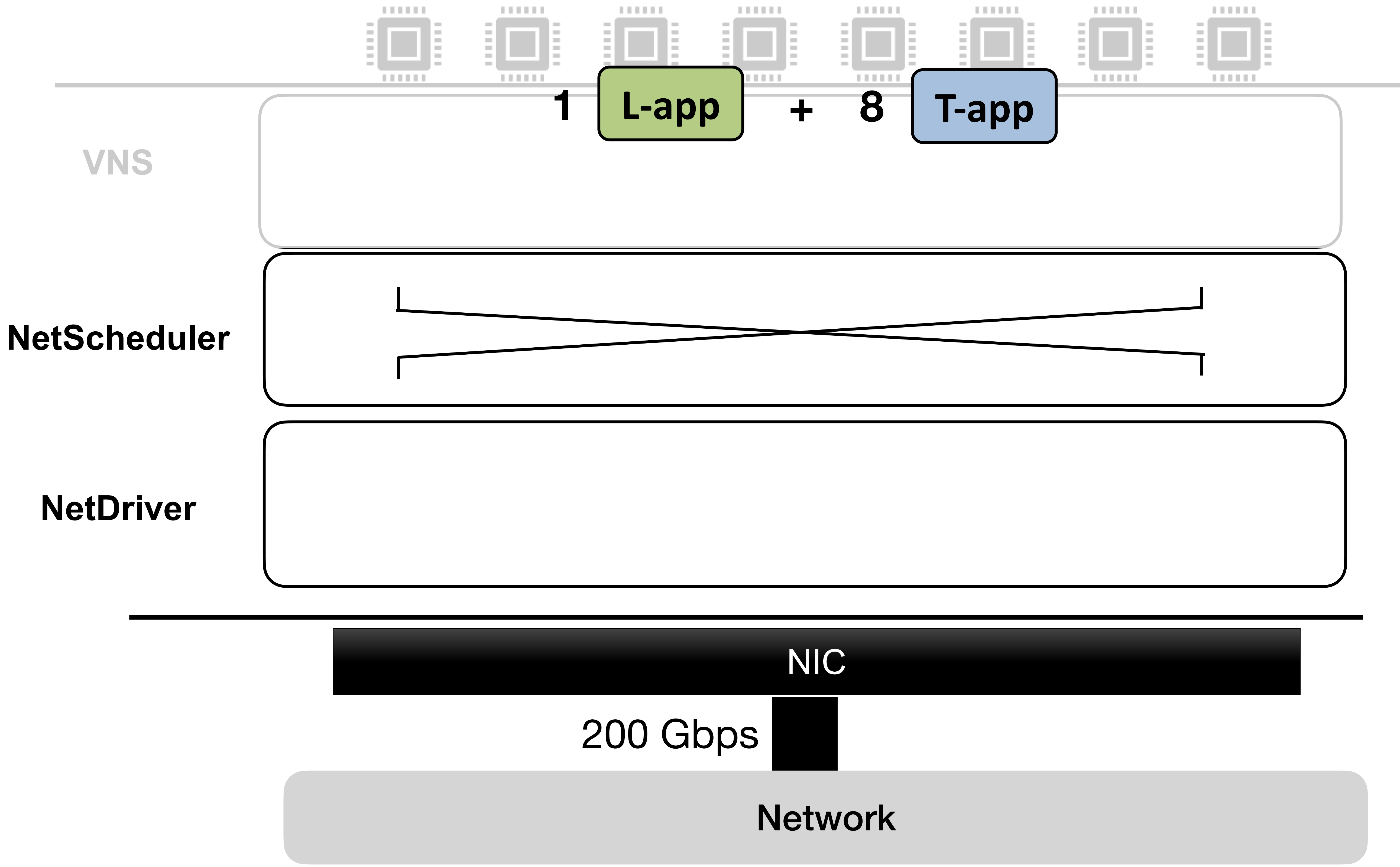
NetChannel enables saturating a Terabit Ethernet link with a single socket

NetChannel enables linear scalability of throughput for short messages

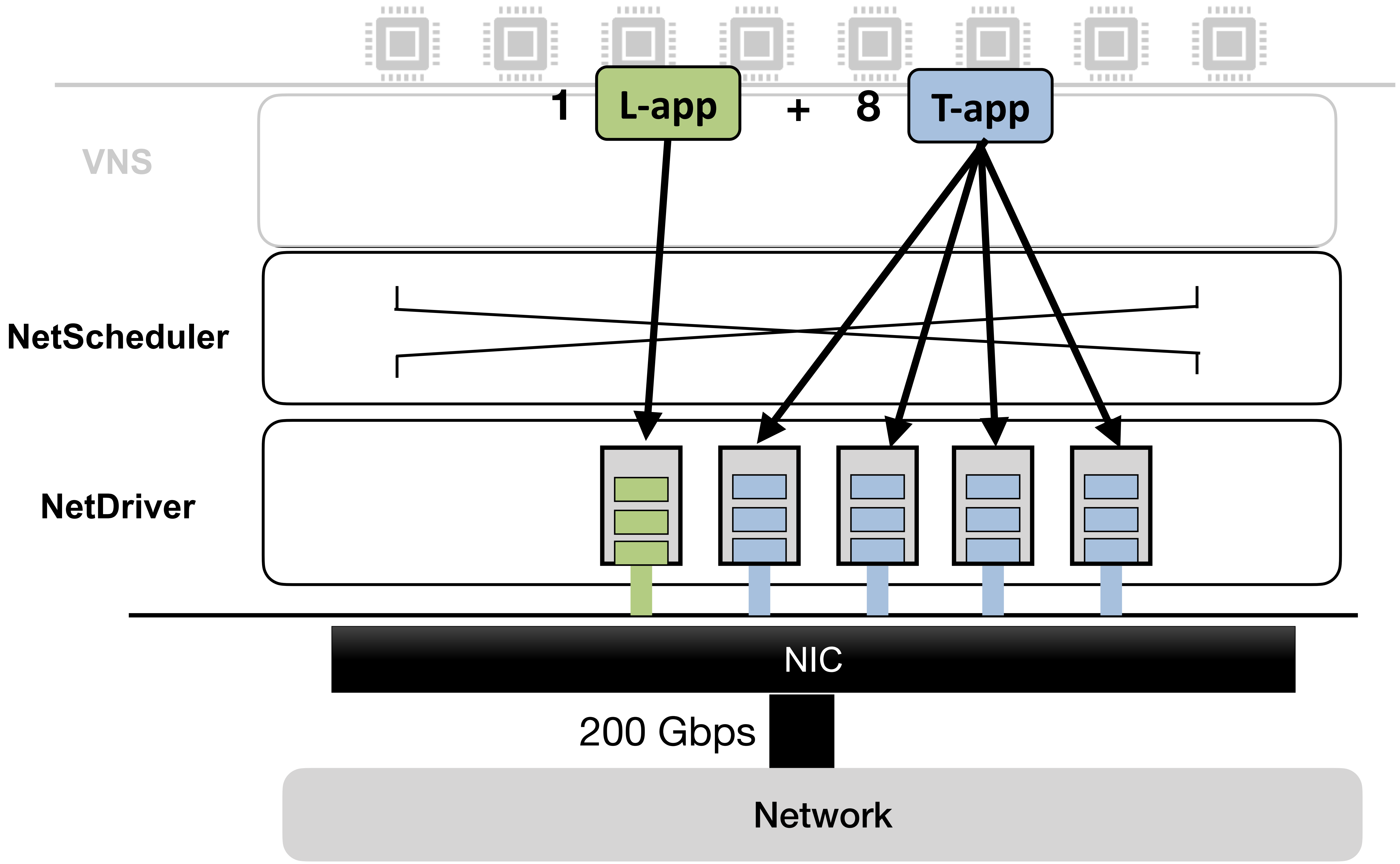
NetChannel: Towards μs Tail Latency



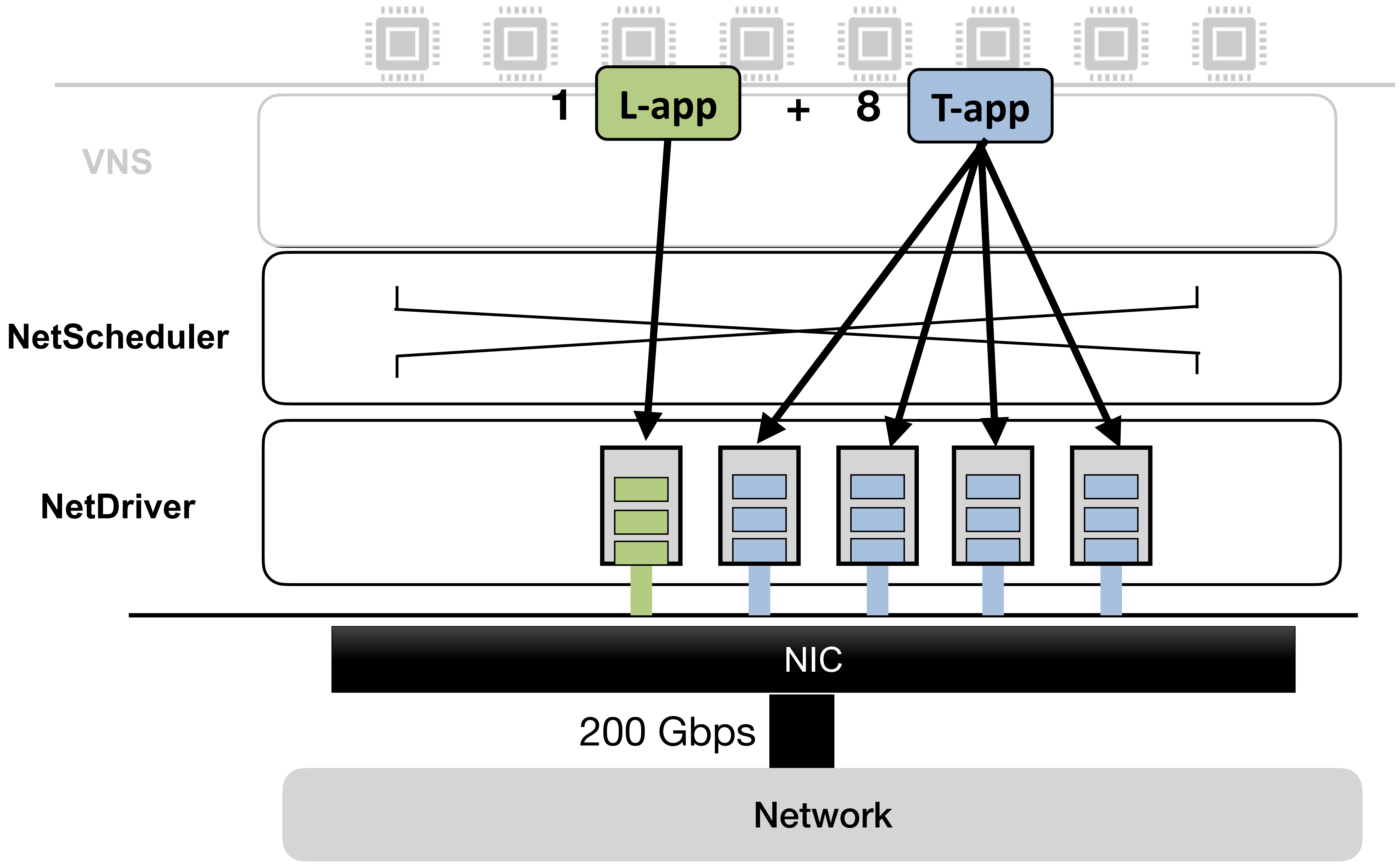
NetChannel: Towards μs Tail Latency

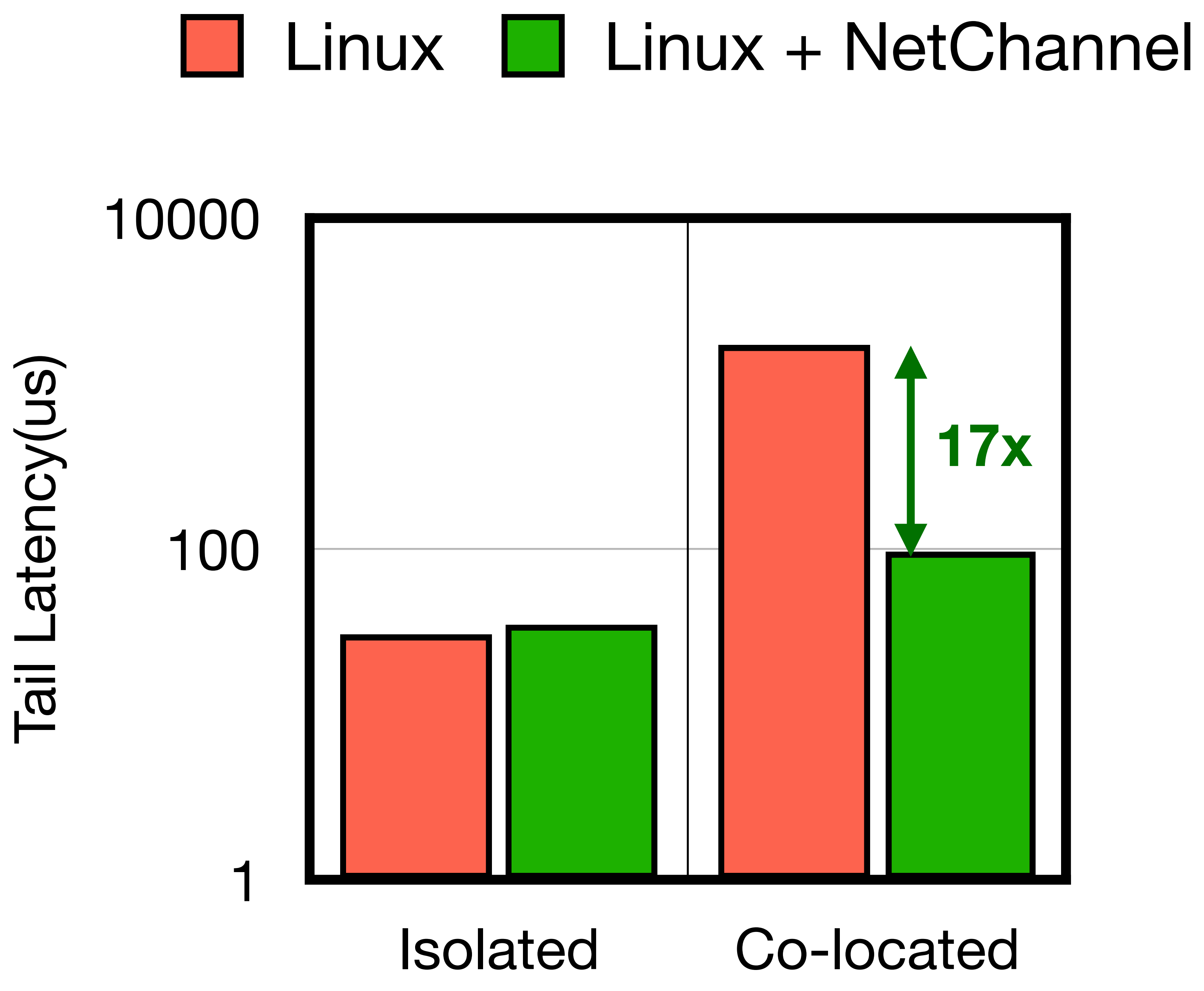


NetChannel: Towards μs Tail Latency



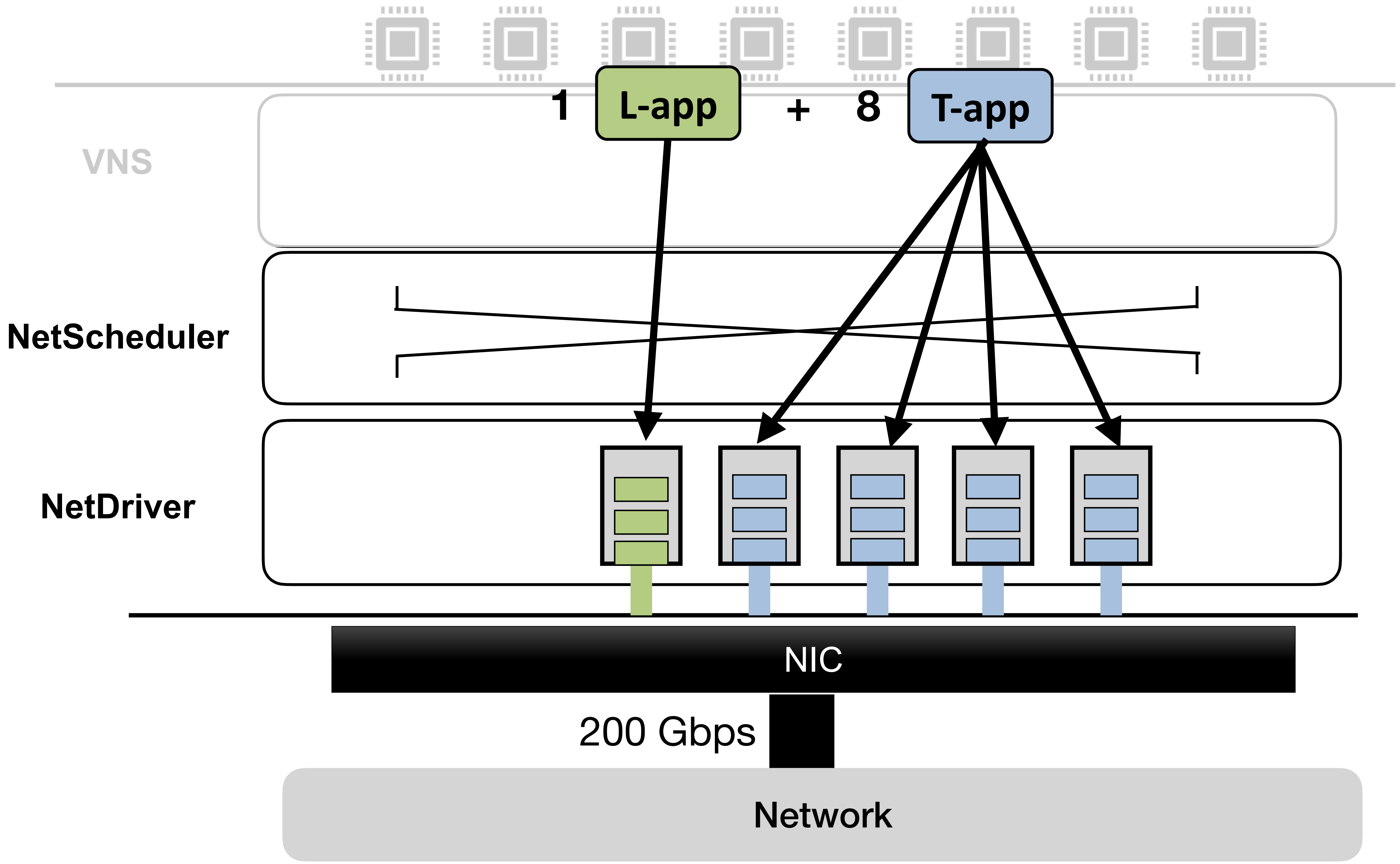
NetChannel: Towards μs Tail Latency

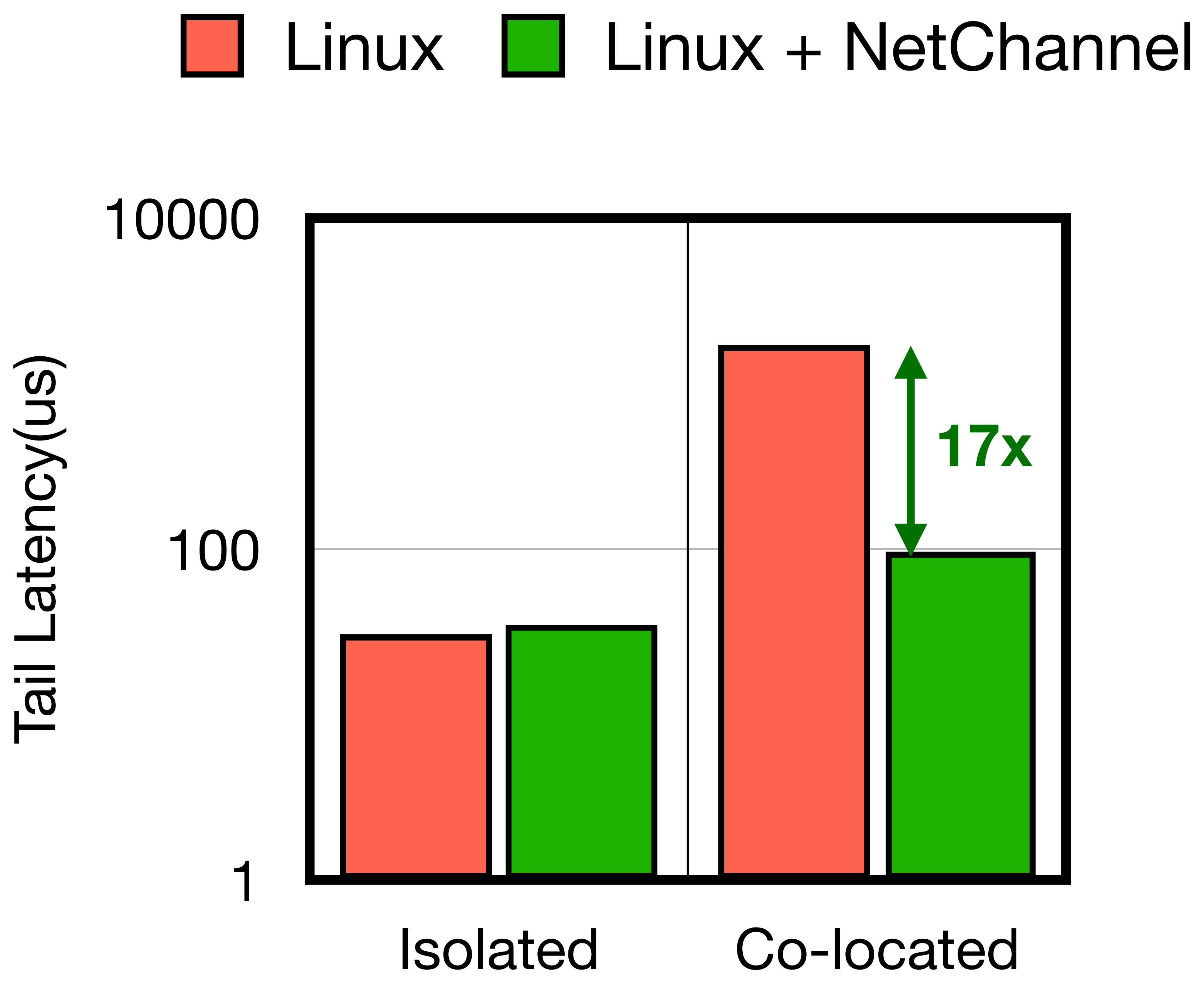




Icon made by Freepik from www.flaticon.com

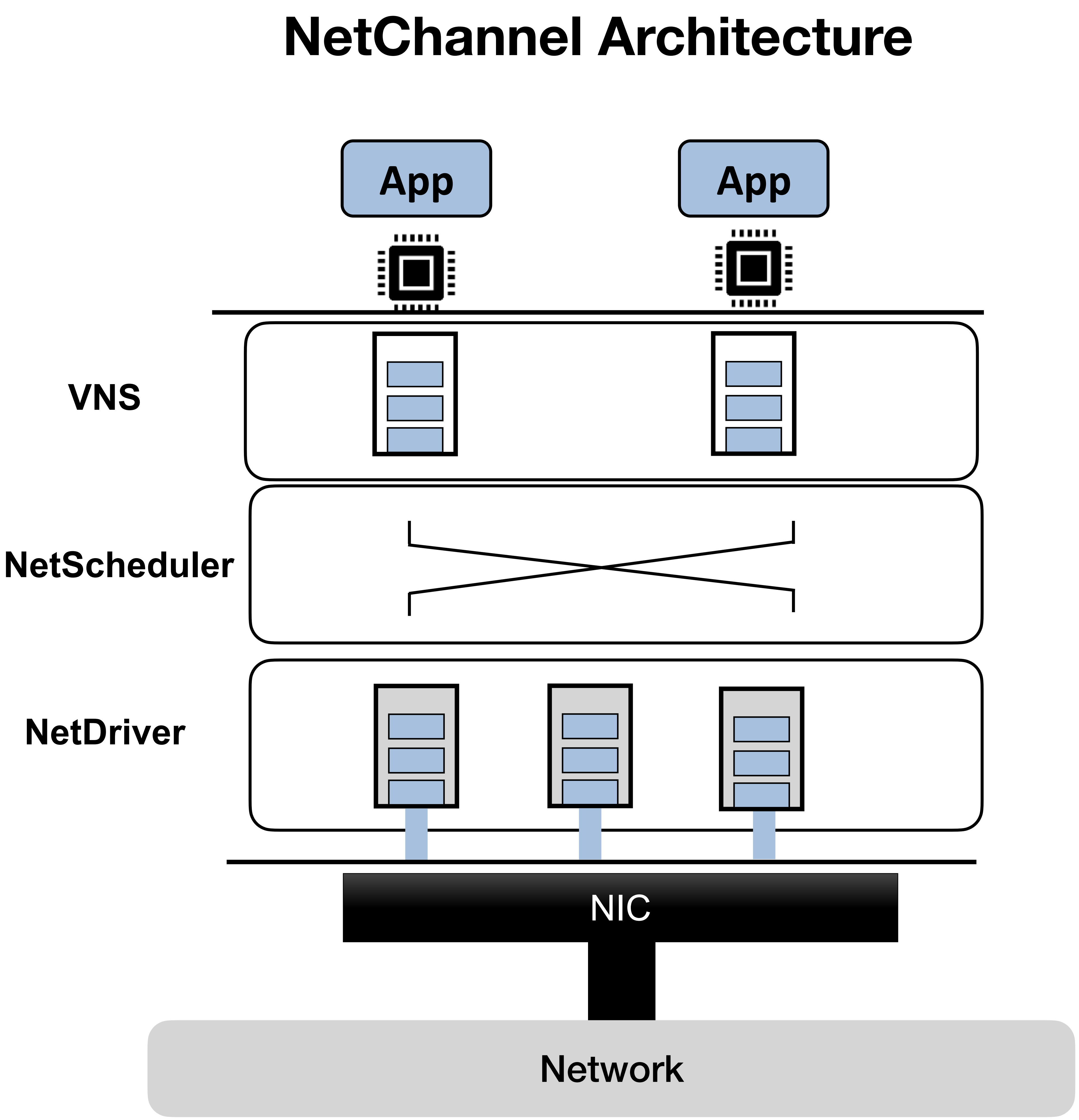
NetChannel: Towards µs Tail Latency



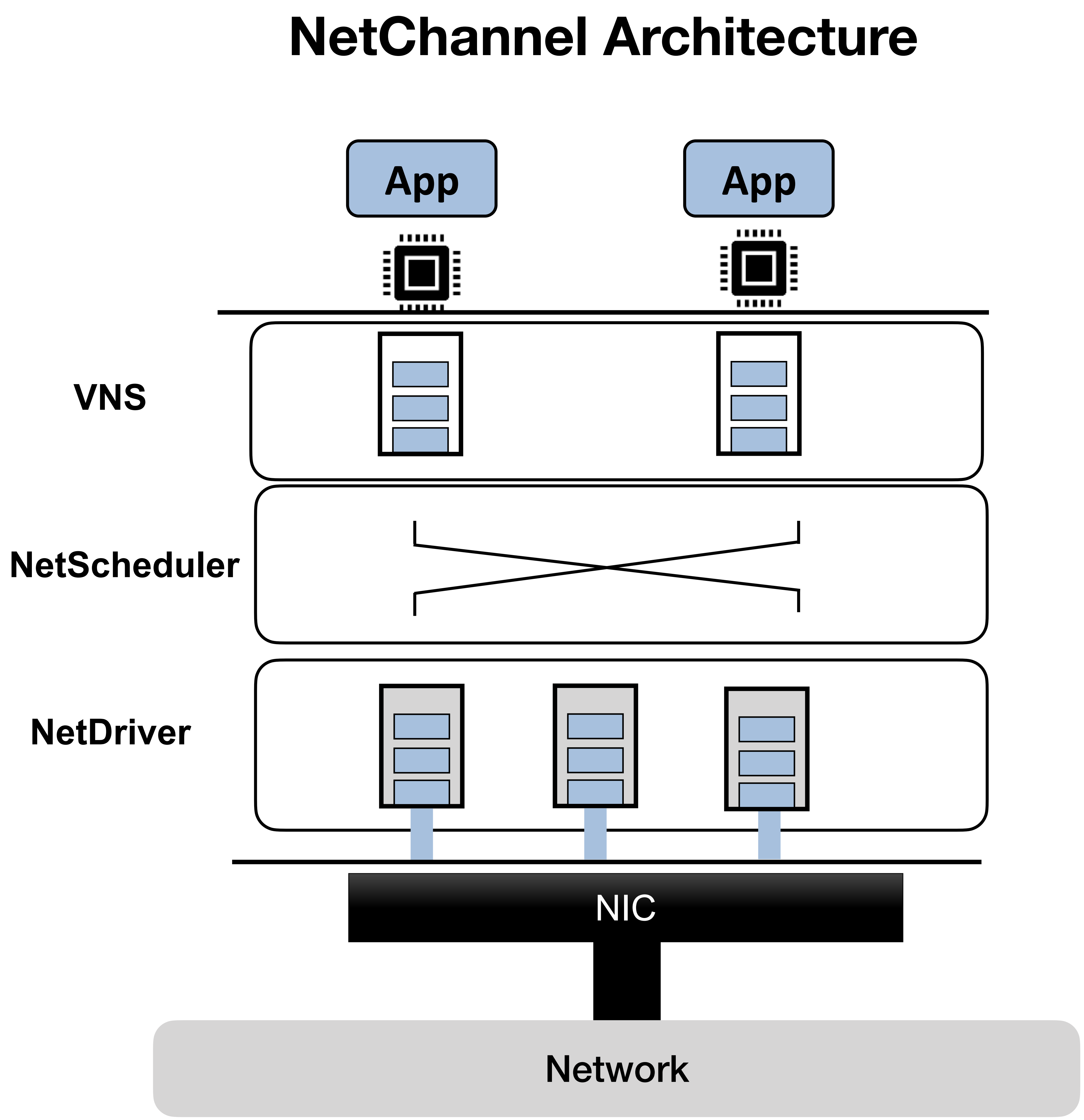


NetChannel enables µs-scale Tail Latency for L-apps even when co-located with T-apps

Future Directions

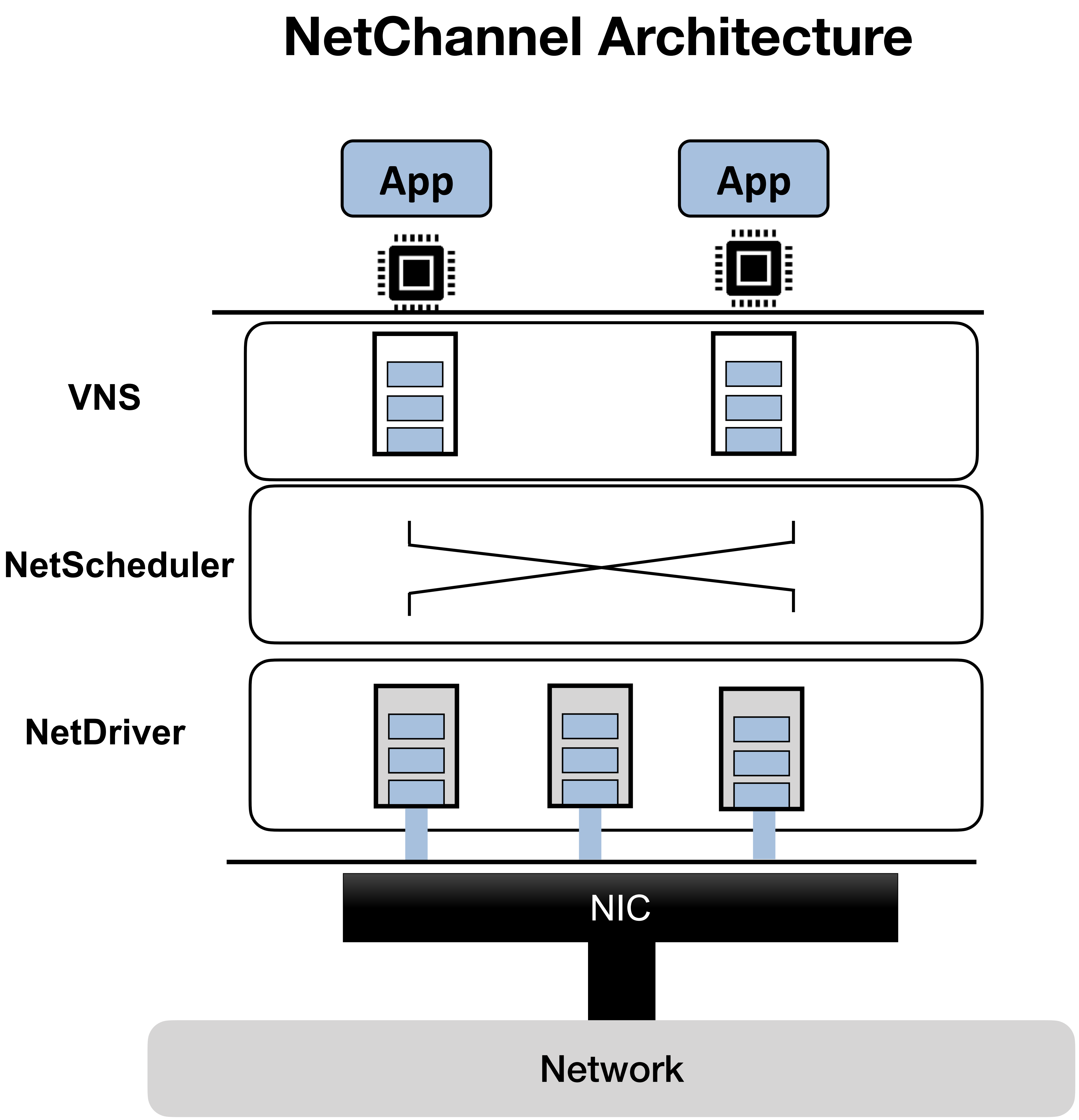


Future Directions



Realizing new NetScheduler policies

Future Directions

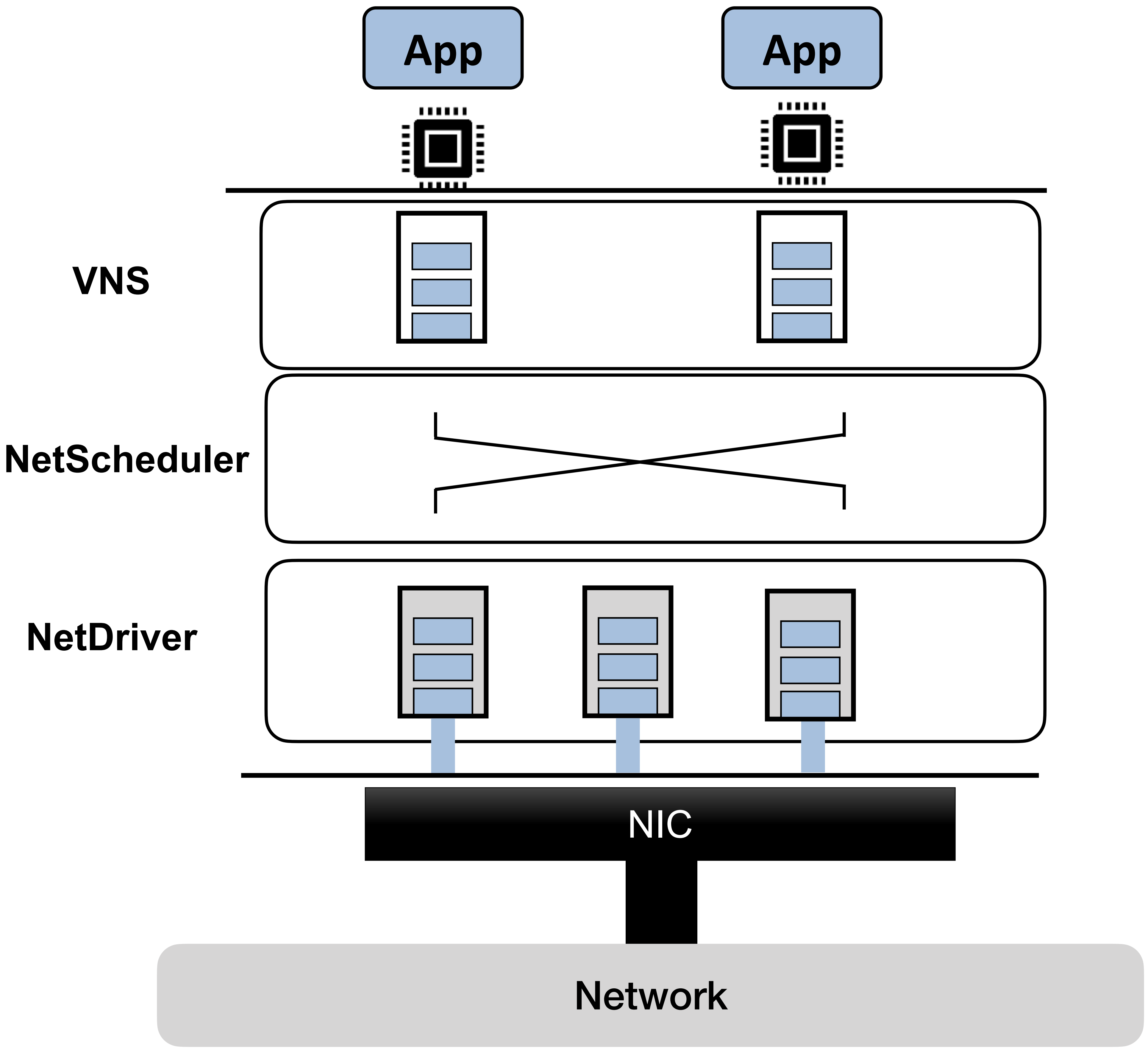


Realizing new NetScheduler policies

Integrating new transport protocols

Future Directions

NetChannel Architecture



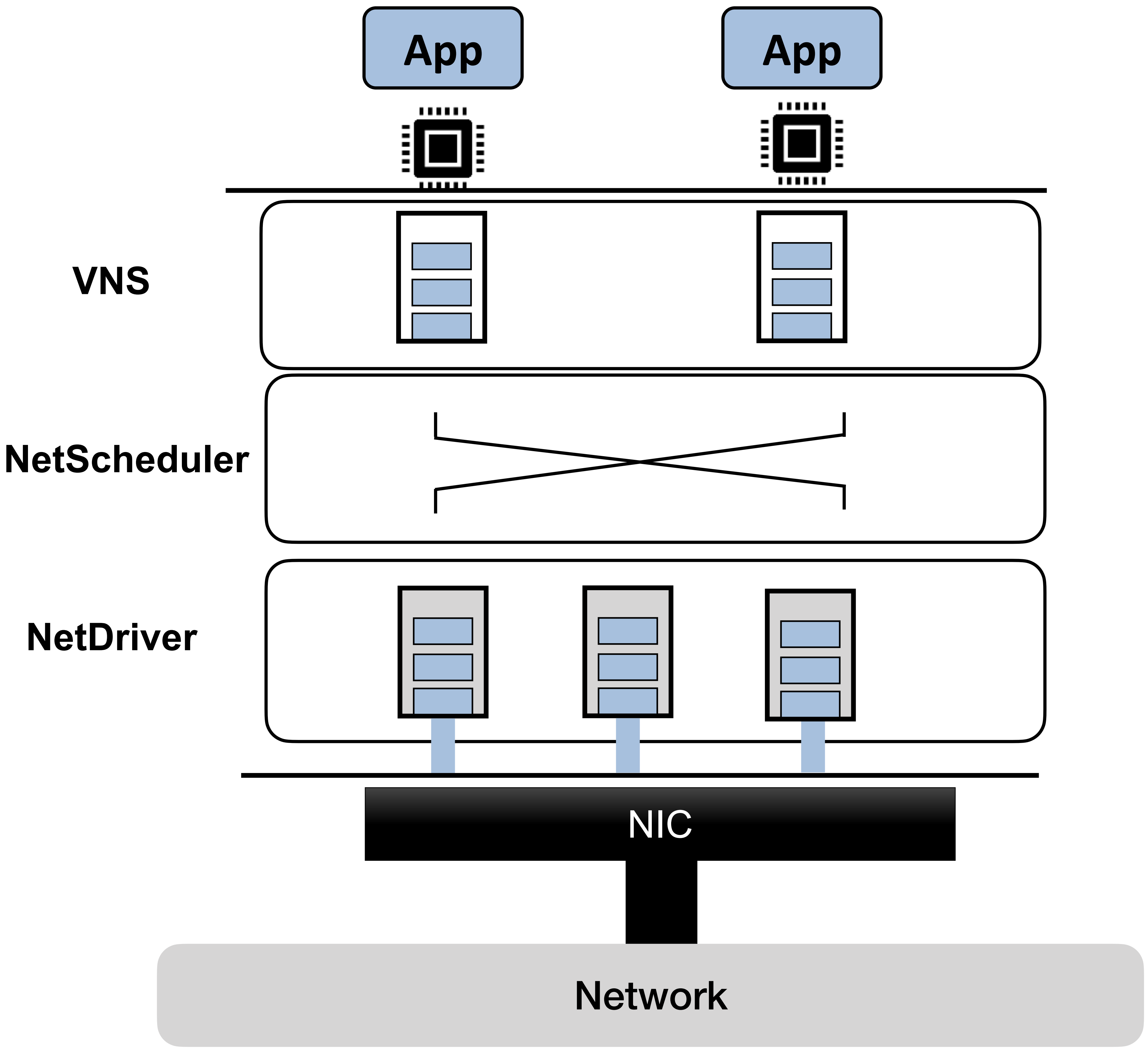
Realizing new NetScheduler policies

Integrating new transport protocols

Realizing NetChannel in userspace / hardware

Future Directions

NetChannel Architecture



Realizing new NetScheduler policies

Integrating new transport protocols

Realizing NetChannel in userspace / hardware

https://github.com/Terabit-Ethernet/NetChannel