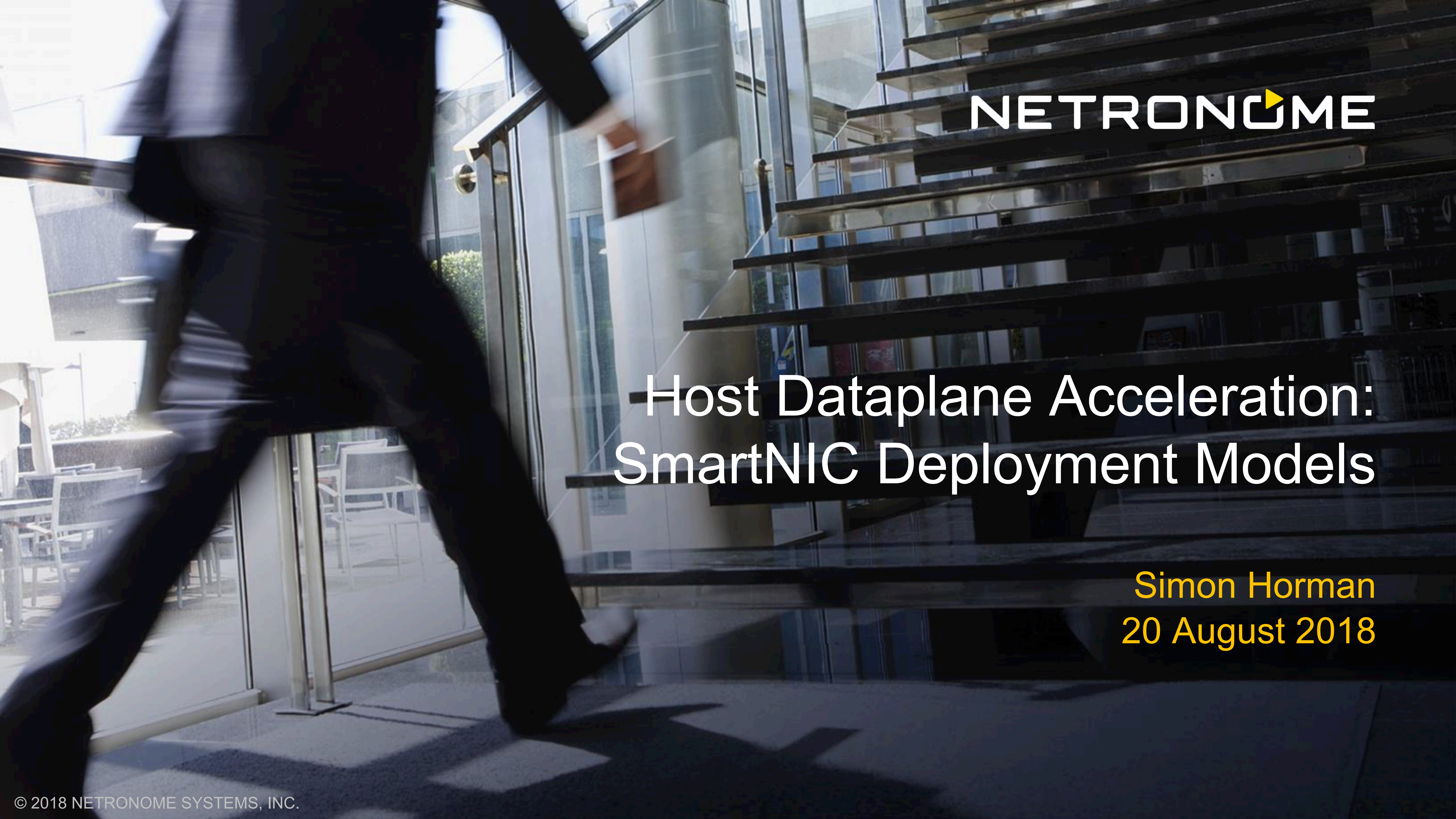


NETRONOME

Host Dataplane Acceleration: SmartNIC Deployment Models

Simon Horman

20 August 2018

Agenda

- Introduction
  - Hardware and Software Switching
  - SDN Programmability
  - Host Datapath Acceleration
- Acceleration Models
  - Existing Host Datapath Acceleration
  - Extended Datapath Acceleration
  - New Datapath Acceleration
- Closing Remarks

NETRONOME

Hardware and Software Switching

Historically Switching Done in Hardware

- Dedicated physical switch
- Fast
- Fixed feature set
- Inflexible

Virtualisation Moved Switching to Host

- Desire to share network between hosts and VMs
- Software bridging or user-space networking used to achieve this
- Flexible
- Slow

SR-IOV Moved Hardware Switching to the Host

- Embedded switch between
  - Physical Port
  - “PF” network interfaces used by host
  - VF network interfaces used by VMs
- Performance improvement over software based switching to VMs
- Reduced flexibility

SDN Moved Host Switching Back into Software

- Exploited flexibility of software
- Many features emerged: f.e.:
  - Tunnel Termination
  - Rich flow key for matching on L2 ~ L4
  - Packet modification possible: set packet header fields
  - Stateful Security Policies (Conntrack)
  - NAT
  - Increasing programmability
- Slow/consumes CPU resources

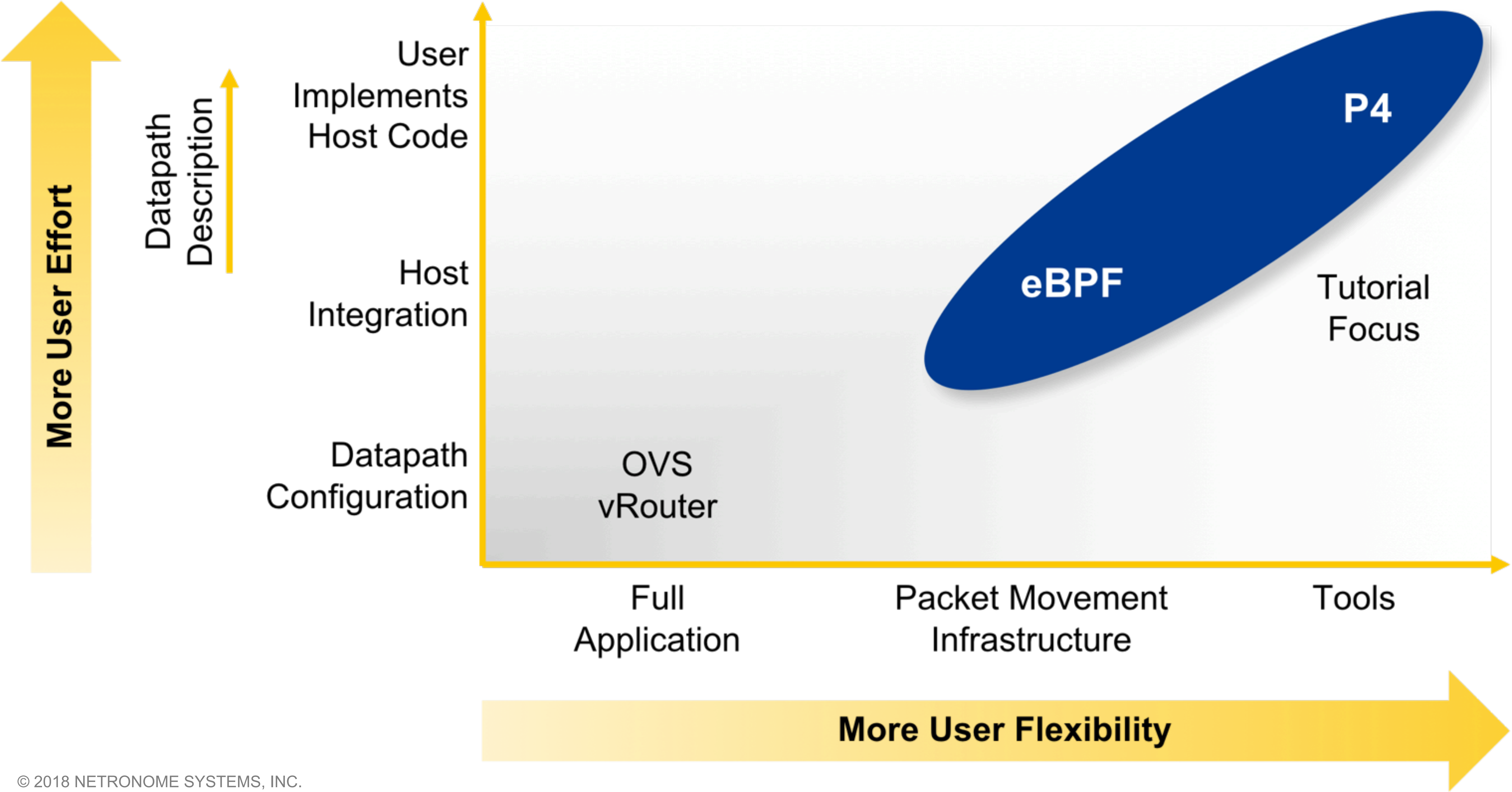
NETRONOME

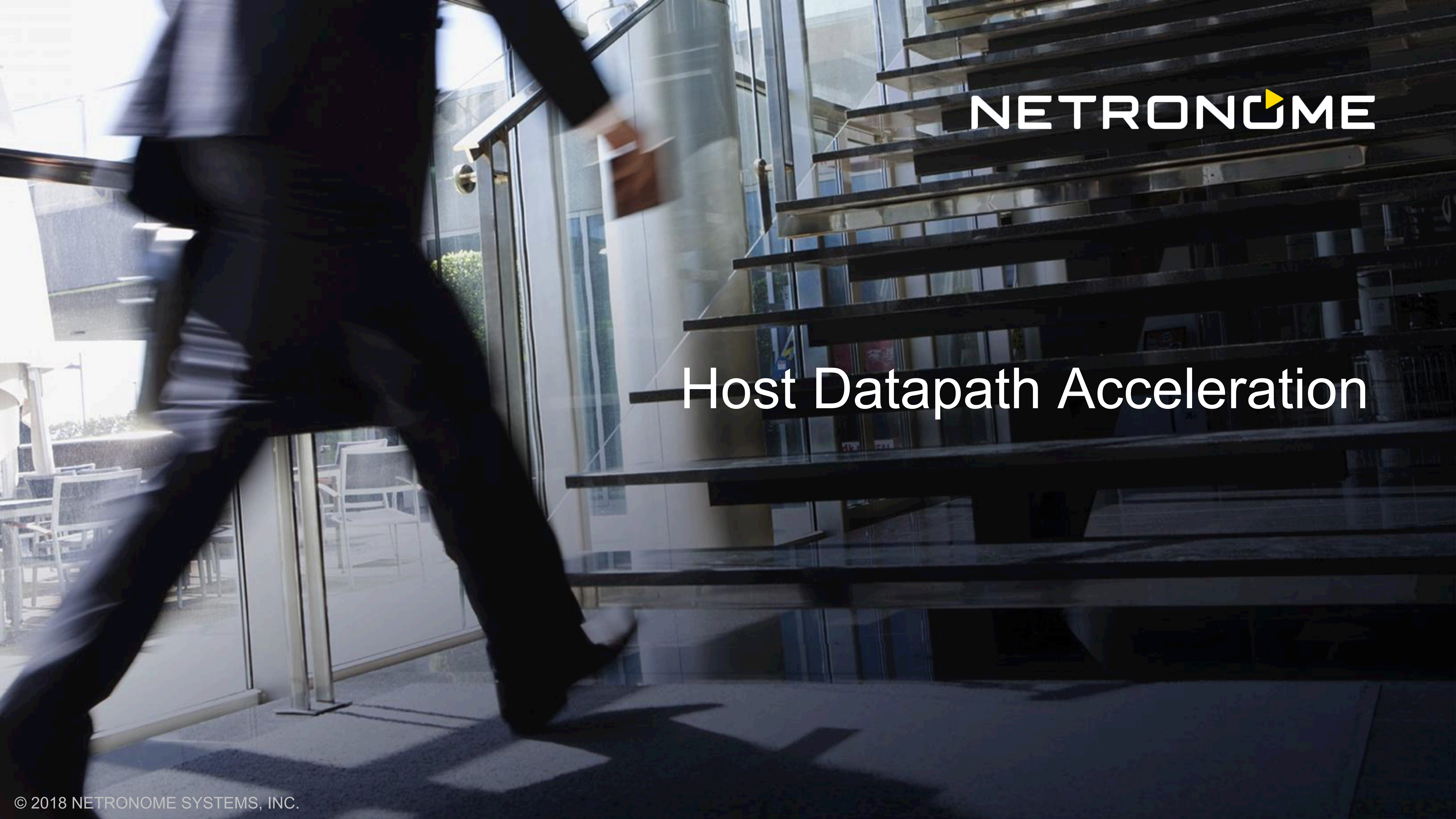
Datapath Programmability

SDN Programmability

- Applications
  - Rich feature set but fixed function
  - Policy implemented at run-time, f.e. using match/action tables
  - Software is malleable, but new features require modification of application
  - f.e.: Open vSwitch, vRouter
- Packet Movement Infrastructure
  - Allows programing of part of datapath
  - With fallback option
  - f.e.: eBPF
- Tools
  - Allows full description of datapath
  - f.e.: P4

NEW Tools vs. Applications



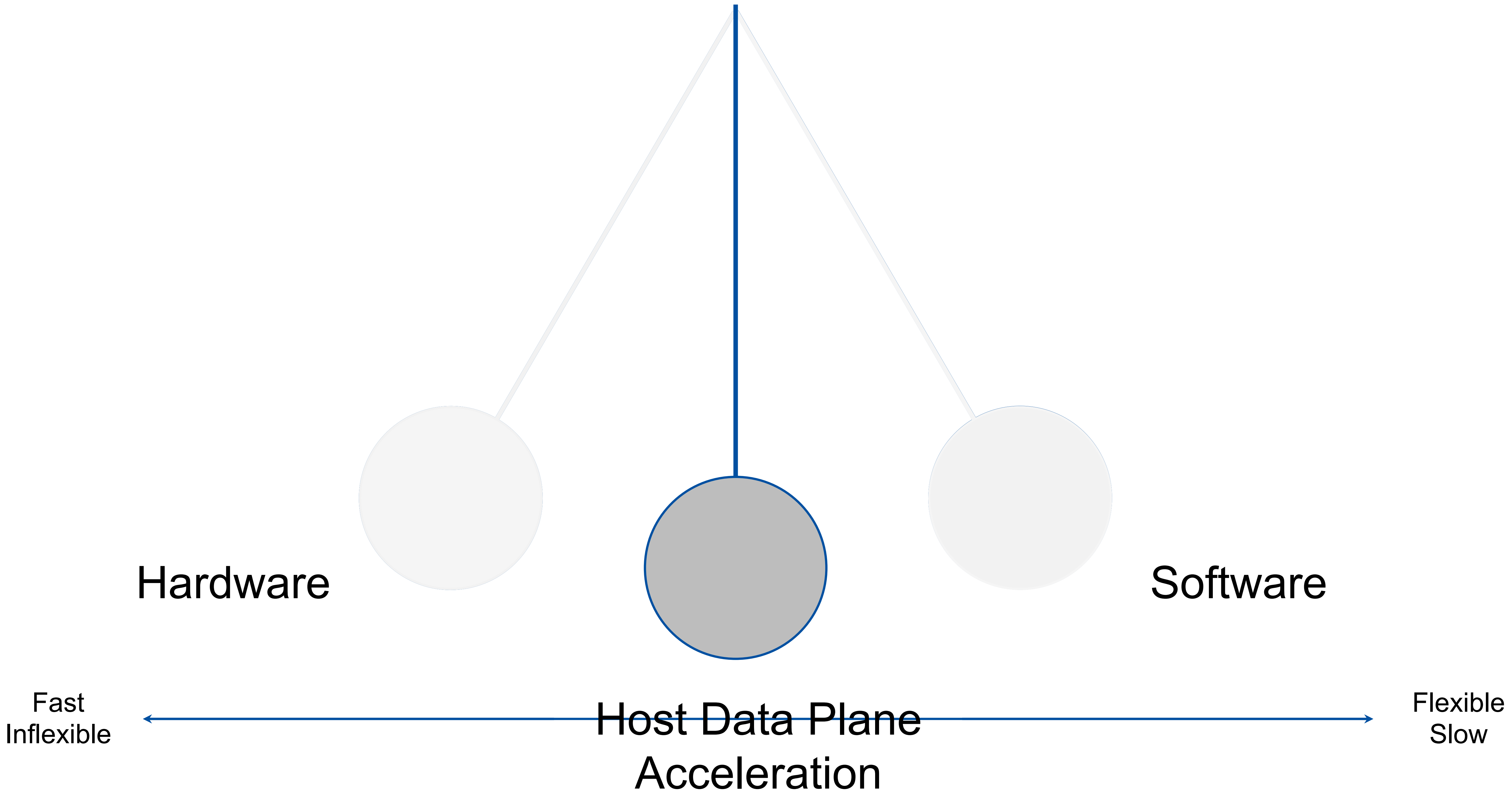


NETRONOME

Host Datapath Acceleration

Host Datapath Acceleration

- Combine flexibility of software with performance of hardware



Pendulum motive credit: Rashid Khan, “Red Hat’s perspective on OVS HW Offload Status”, presented at Open vSwtich 2018 Conference

Comprehensive and Proven SmartNIC Platform

Basic NIC Features

Offload NIC Features

Programmable NIC Features

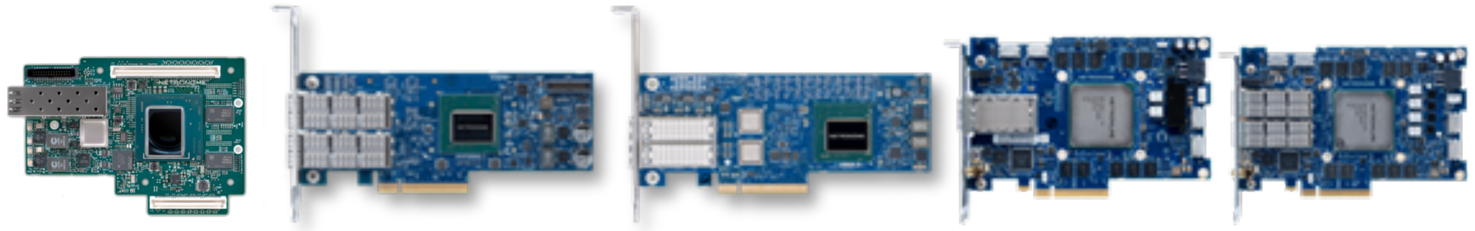
Data Plane Programming Tools for Custom Feature Adds
P4, C, eBPF/XDP-based Programming Layer

Virtual Switching and Routing	VXLAN, MPLS, MPLS over GRE	ACLs and Security Groups	vProbes, In-band Telemetry	DDoS and Load Bal	Visibility using SSL-on-a-NIC	Congestion and Tail Latency Reduction

Virtualization Layer and SDN (with standard offload)
Inbox Open vSwitch (OVS) in RHEL 7.5; vRouter in Contrail Cloud 3.x/4.x

Basic NIC Upstreamed Linux Device Drivers
Stateless Offloads, SR-IOV, DPDK, Express Virtio (XVIO)

Agilio SmartNIC Family
10-100Gb/s, w/ 2-8GB DDR, up to 4 ARM Cores



NFP Silicon Family
with 36-120 cores, up to 960 processing threads

NETRONOME

Existing Open vSwitch Datapath Acceleration

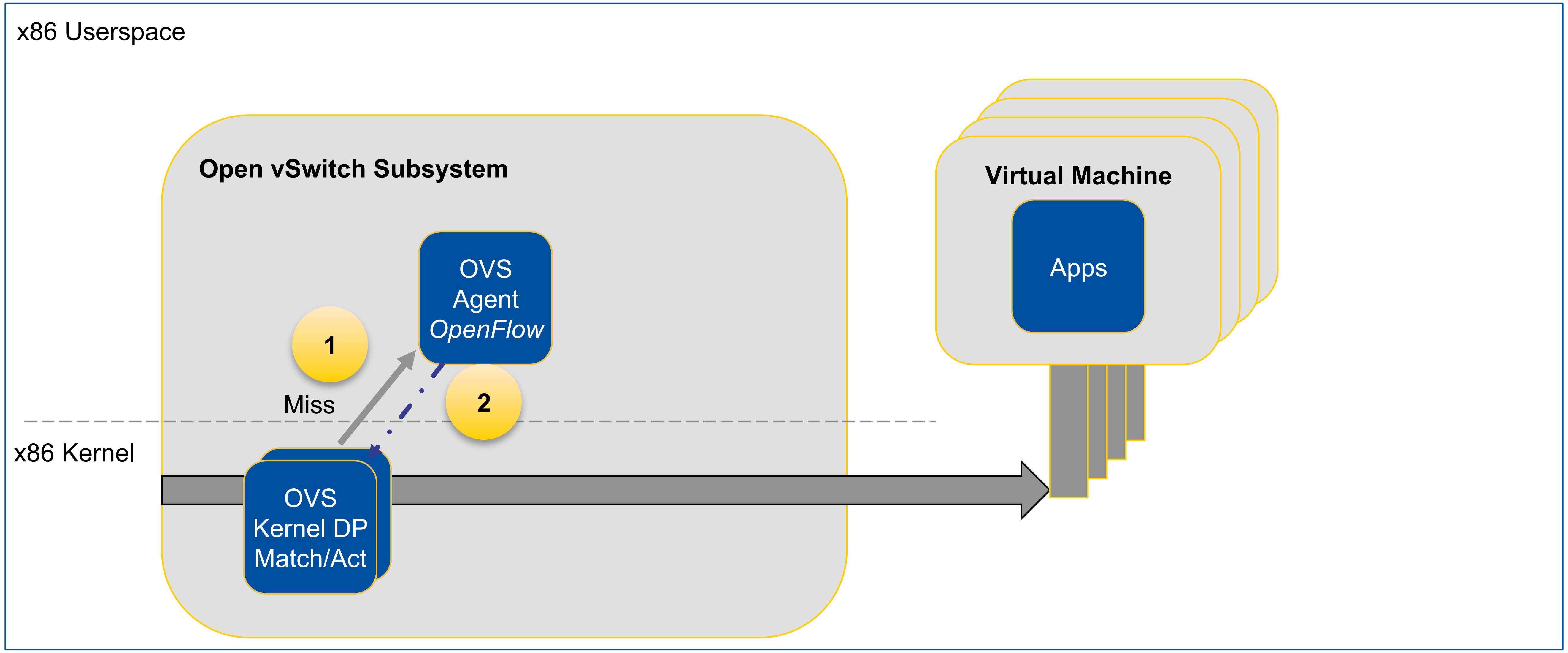
What is Open vSwitch (OVS)?

- Production quality multilayer virtual switch
- Widely used in conjunction with OpenStack
- Allows policy to be describe using OpenFlow match/action tables

OVS Datapath

1 Flow table miss

2 OVS userspace agent populates kernel cache; instructs OVS kernel DP to execute packet



Agilio® OVS Datapath Acceleration

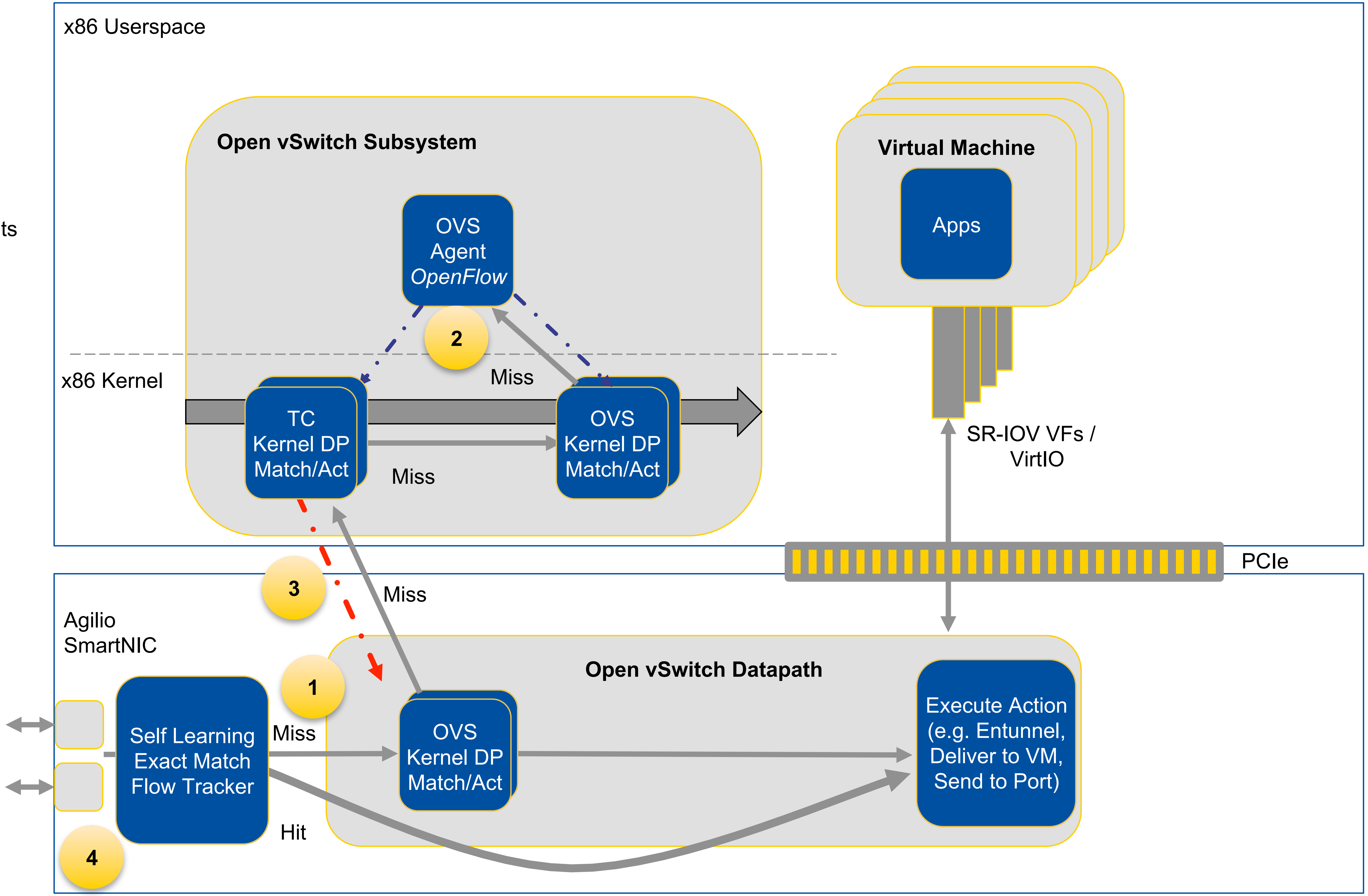
NETRONOME

1 Flow table miss

2 OVS userspace agent populates kernel cache; instructs OVS kernel DP to execute packet

3 Offload datapath: copy match tables, sync stats

4 Flow tracking: per-microflow state learning



NETRONOME

Existing eBPF Datapath Acceleration

What is eBPF?

eBPF is a simple way to extend the functionality of the kernel at runtime

- Small kernel based machine
  - 10 64bit registers
  - 512 byte stack
  - Data structures known as maps (unlimited size)
  - 4K BPF instructions (Bytecode)
- Verifier to ensure kernel safe
  - no loops, not more than 4K insns, not more than 64 maps etc…
- Can be JITed to ensure maximum performance

Used Within Hyperscale-Not a Toy!

NETRONOME

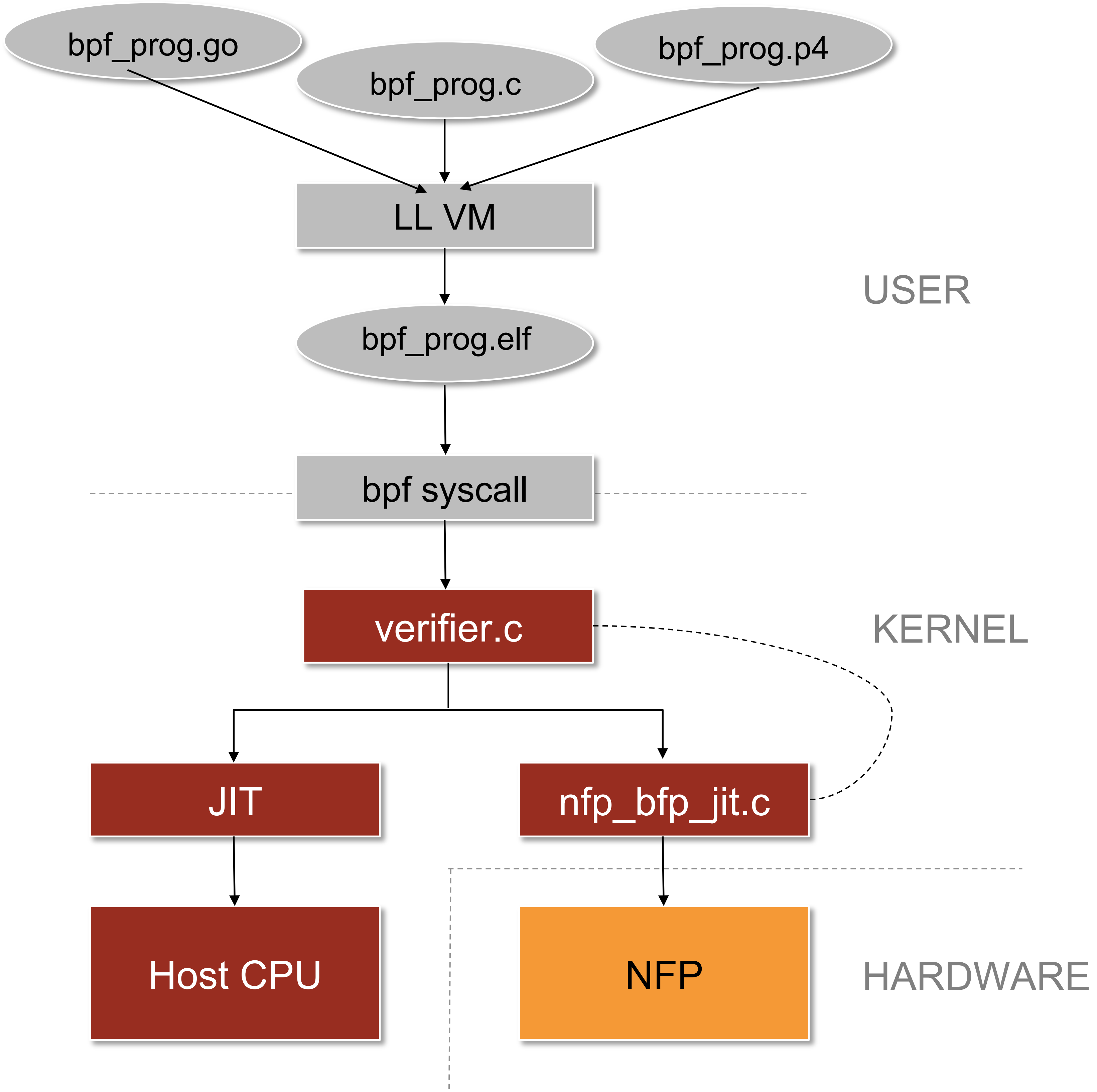
Those who have publicly stated they are using BPF or planning to use BPF include:

- Facebook-Load Balancing, Security
- Netflix-Network Monitoring
- Cilium Project
- Cloudflare-Security
- OVS-Virtual Switching

Due to its upstream safety and kernel support BPF provides a safe, flexible and scalable networking tool.

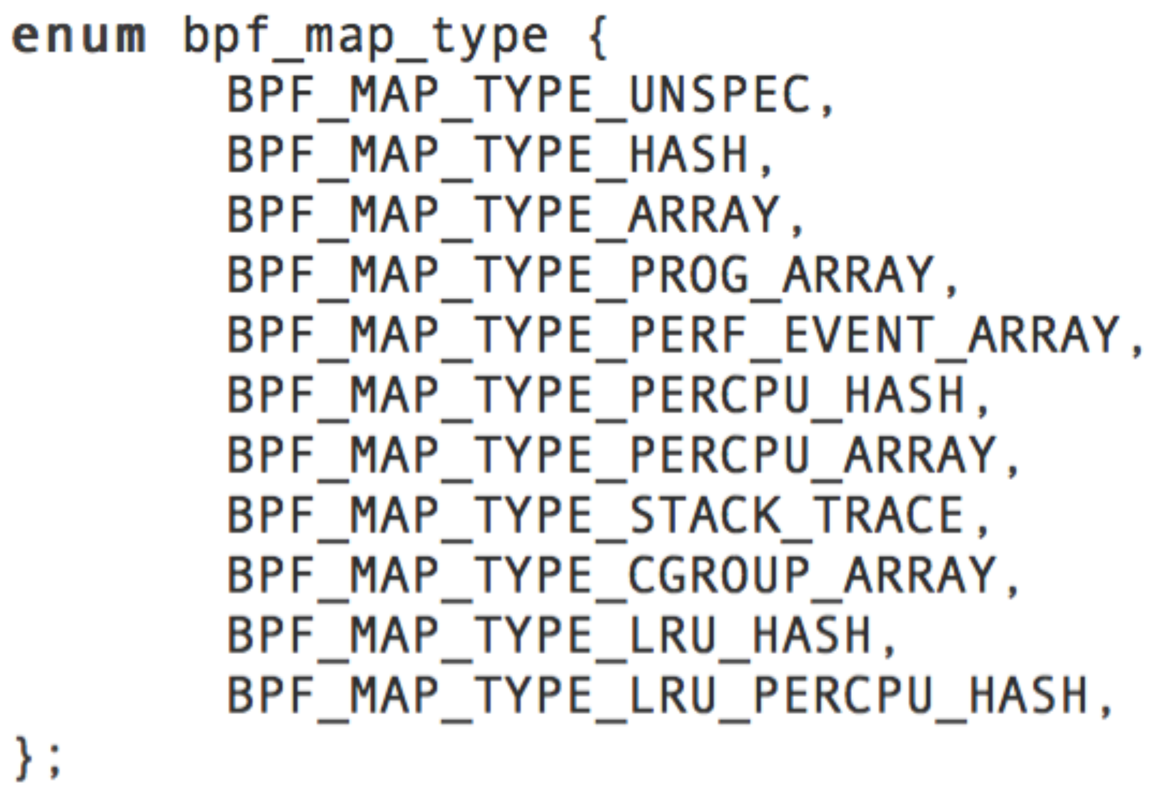
The Programming Model

- LLVM is used to compile from supported languages
  - C, Go, P4
- When Programs are loaded
  - Verifier is called-ensure safety
  - Program is JITed-ensure perf
  - Can also be offloaded
    - nfp_bpf_jit upstream

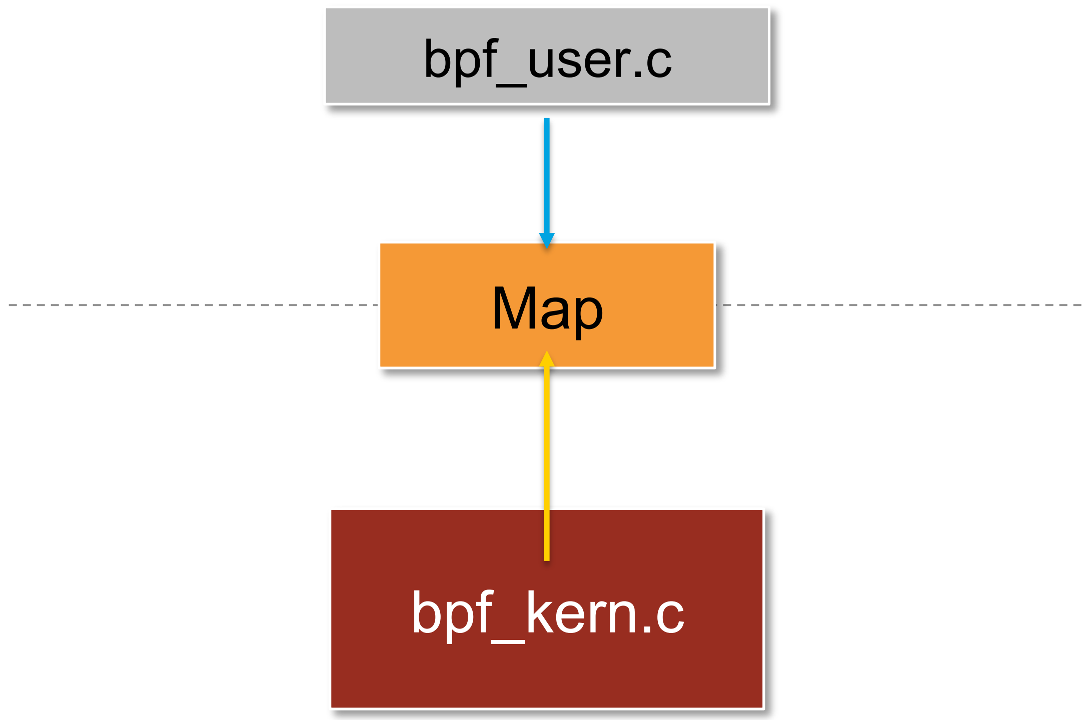


Maps: What They Are

- Maps are key value stores
  - Can be accessed from kernel or user space
  - Used for interaction between kernel and user space programs
- Number of different types of maps
  - Used for interaction between kernel and user space programs



```
enum bpf_map_type {
        BPF_MAP_TYPE_UNSPEC,
        BPF_MAP_TYPE_HASH,
        BPF_MAP_TYPE_ARRAY,
        BPF_MAP_TYPE_PROG_ARRAY,
        BPF_MAP_TYPE_PERF_EVENT_ARRAY,
        BPF_MAP_TYPE_PERCPU_HASH,
        BPF_MAP_TYPE_PERCPU_ARRAY,
        BPF_MAP_TYPE_STACK_TRACE,
        BPF_MAP_TYPE_CGROUP_ARRAY,
        BPF_MAP_TYPE_LRU_HASH,
        BPF_MAP_TYPE_LRU_PERCPU_HASH,
};
```



BPF Kernel Hooks

Many hooks with different purposes

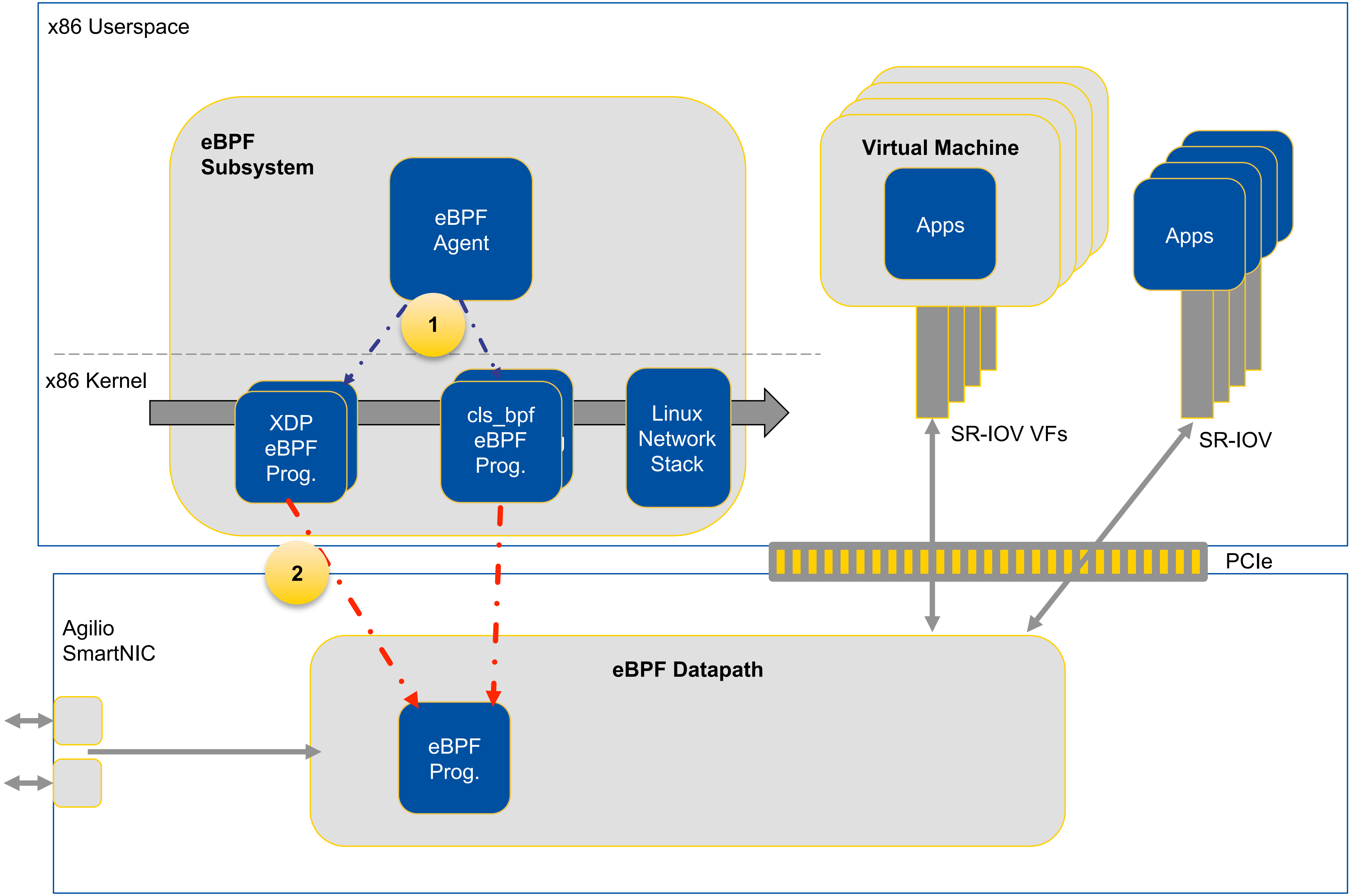
- kprobes
- socket filters-tcpdump-old school!
- seccomp
- netfilter
- TC
- XDP(no skb-super fast!)

Agilio eBPF Acceleration Instantiation

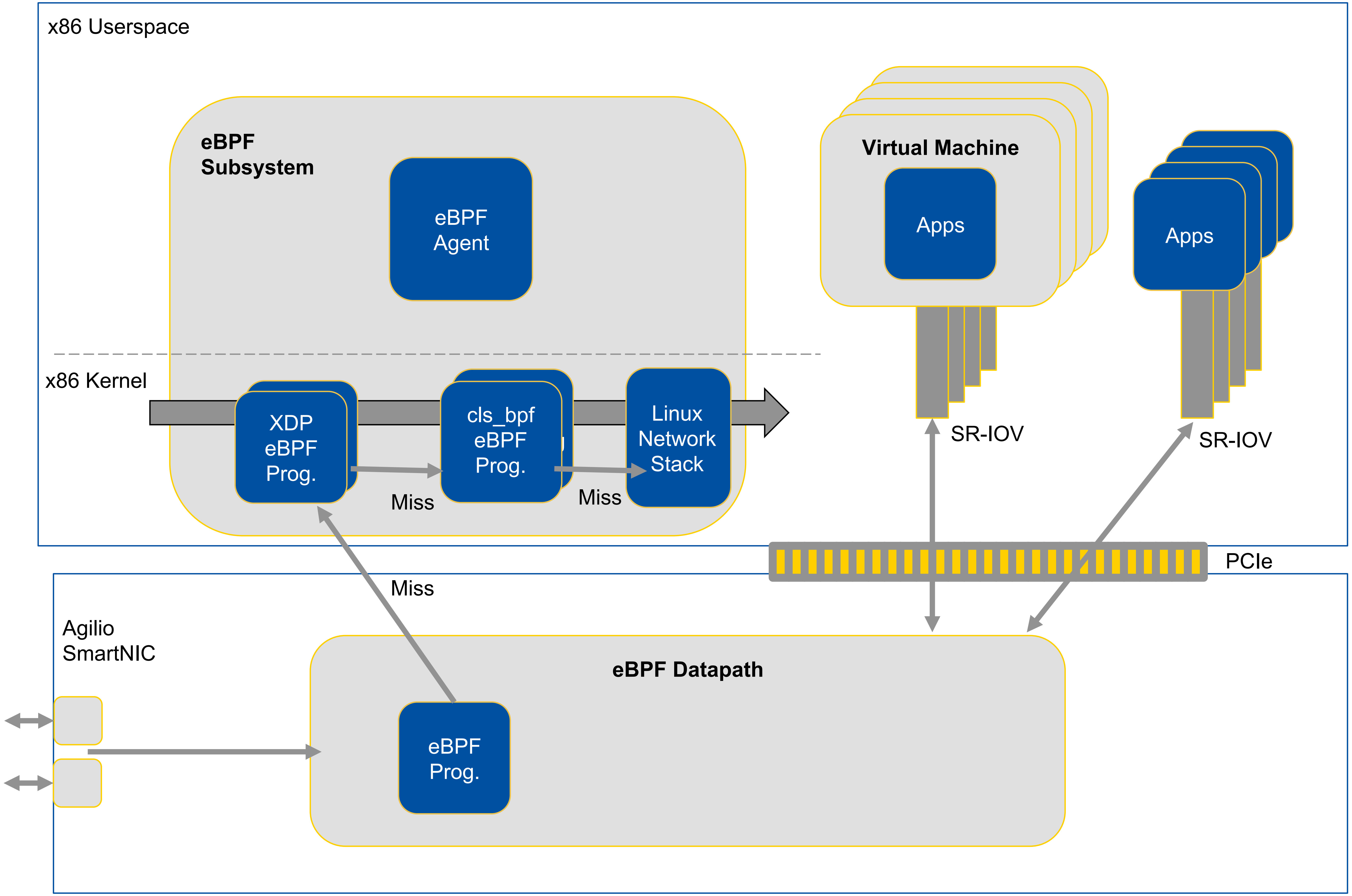
NETRONOME

1 Programs pre-emptively loaded into Kernel

2 JIT program, offload program and maps



Agilio eBPF Acceleration Fallback Path



Offload Existing Datapath: Characteristics

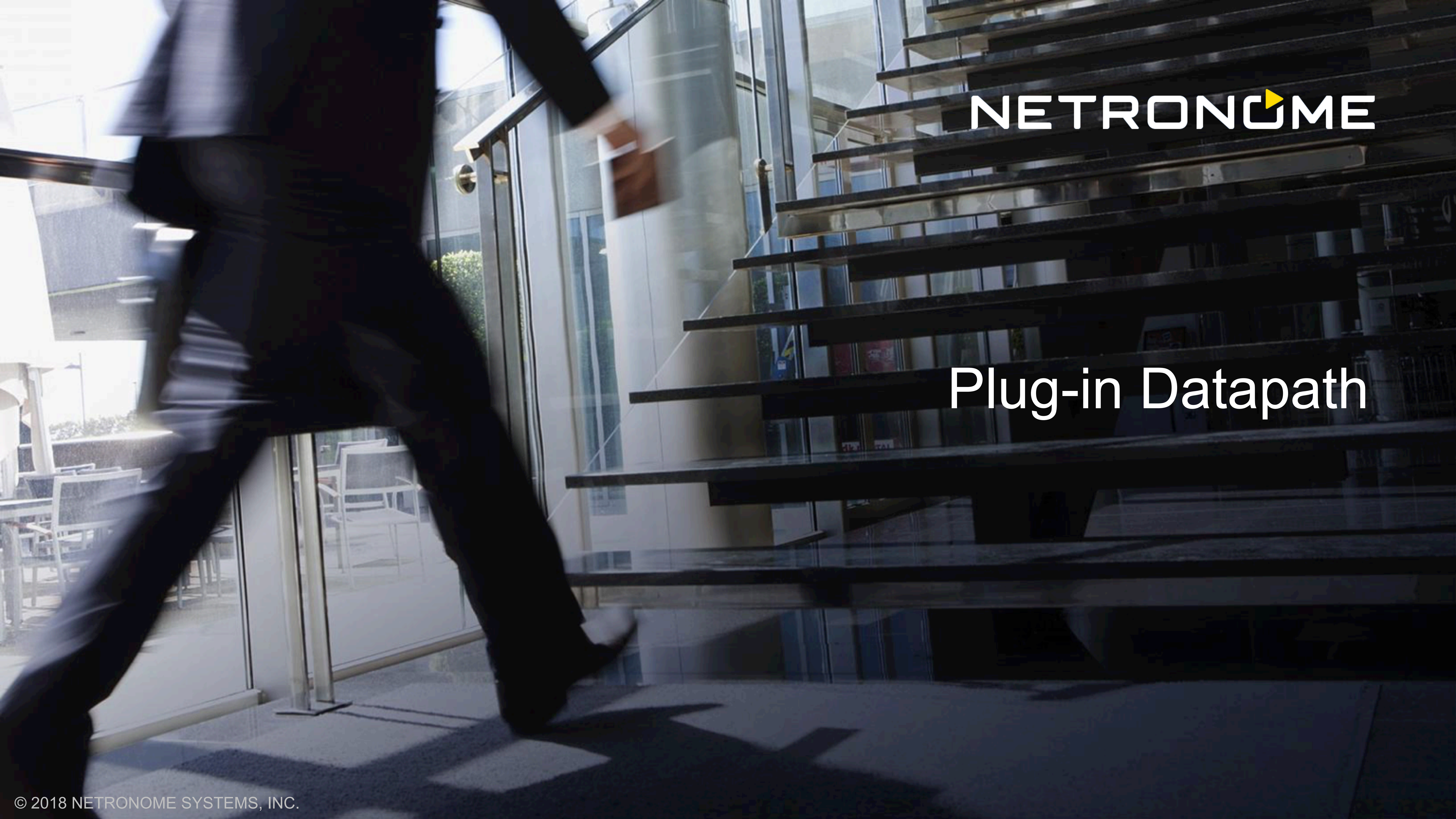
- Flexibility defined by server’s existing datapath software
  - OVS: Configure match/action tables (forwarding/policies)
  - Tungsten Fabric vRouter: Configure forwarding and policies separately
- Integration via drivers/plugins
  - OVS: OpenStack ML2 plugin (with/without SDN controller)
  - OVS: OpenStack driver for OVN
  - TF vRouter: OpenStack driver for TF
- Extend OpenStack to support new concept — SR-IOV path directly to VM while offloading virtual switching to NIC

NETRONOME

Extended and New Datapath Acceleration

What is P4?

- P4 is a domain specific language for describing datapaths
- Description may be compiled into datapath



NETRONOME

Plug-in Datapath

Extending OVS using P4/C Plugins

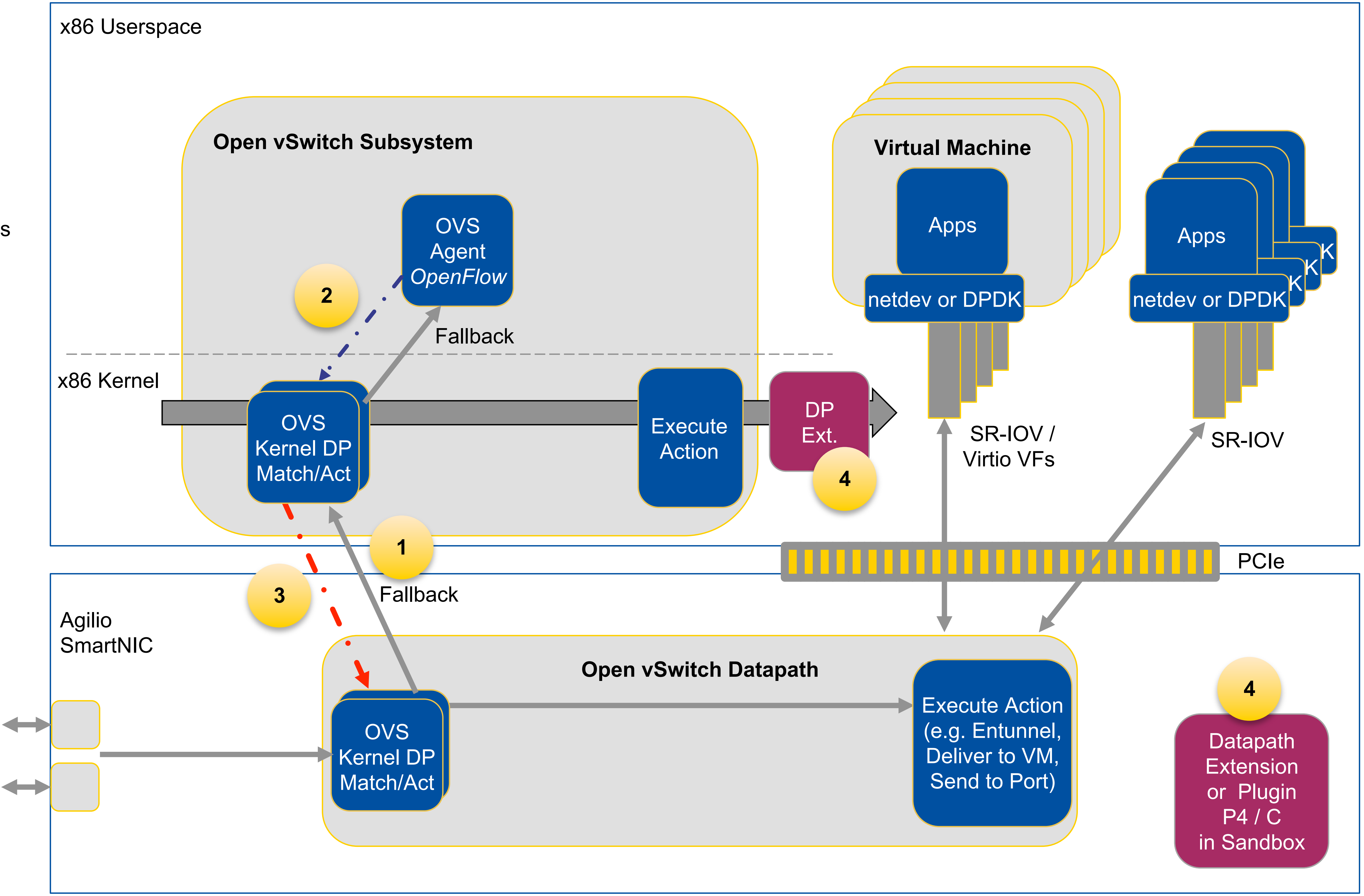
NETRONOME

1 Flow table miss

2 OVS userspace agent populates kernel cache; instructs OVS kernel DP to execute packet

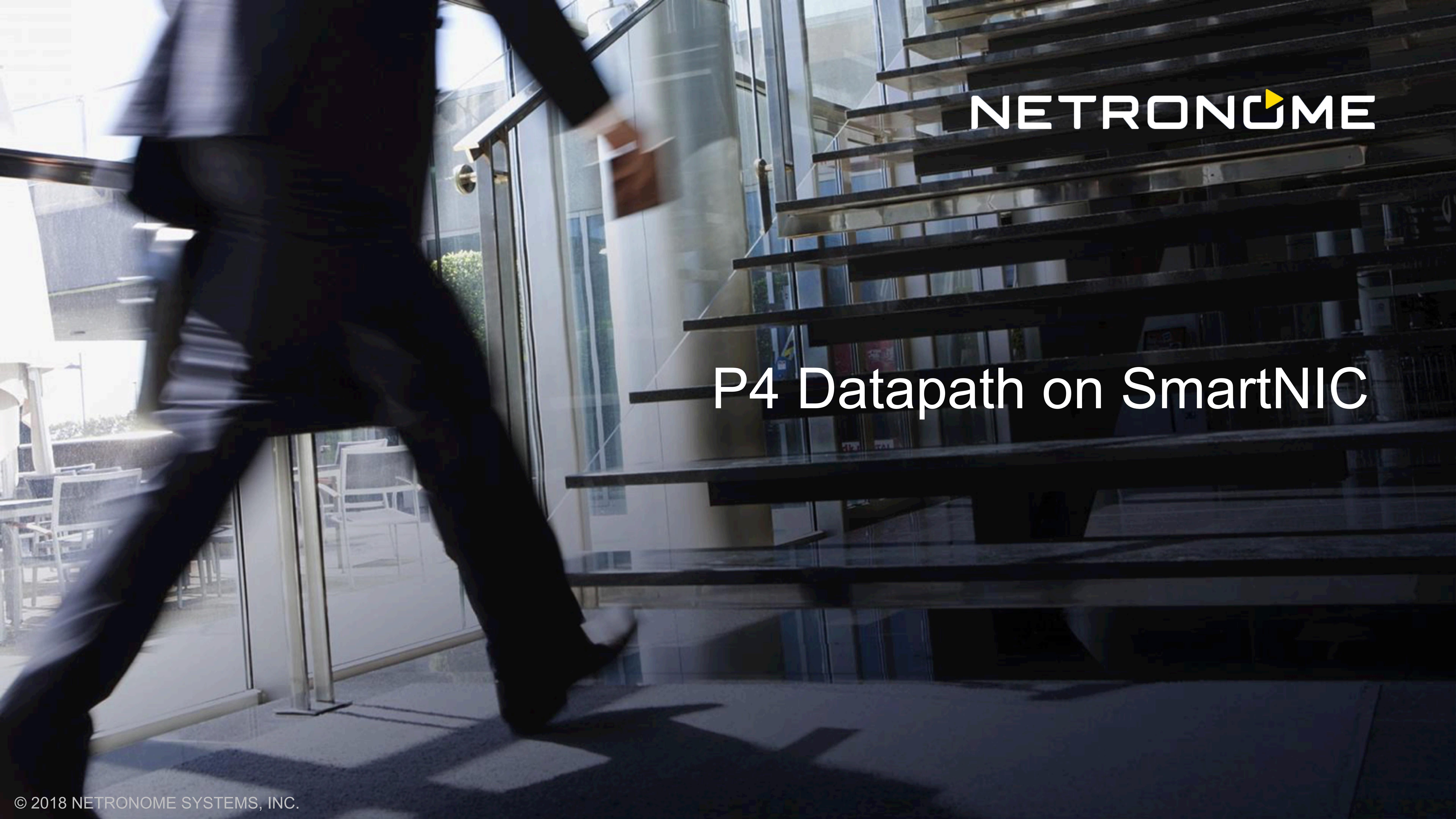
3 Offload datapath: copy match tables, sync stats

4 Datapath extension software



Plugin in Datapath: Characteristics

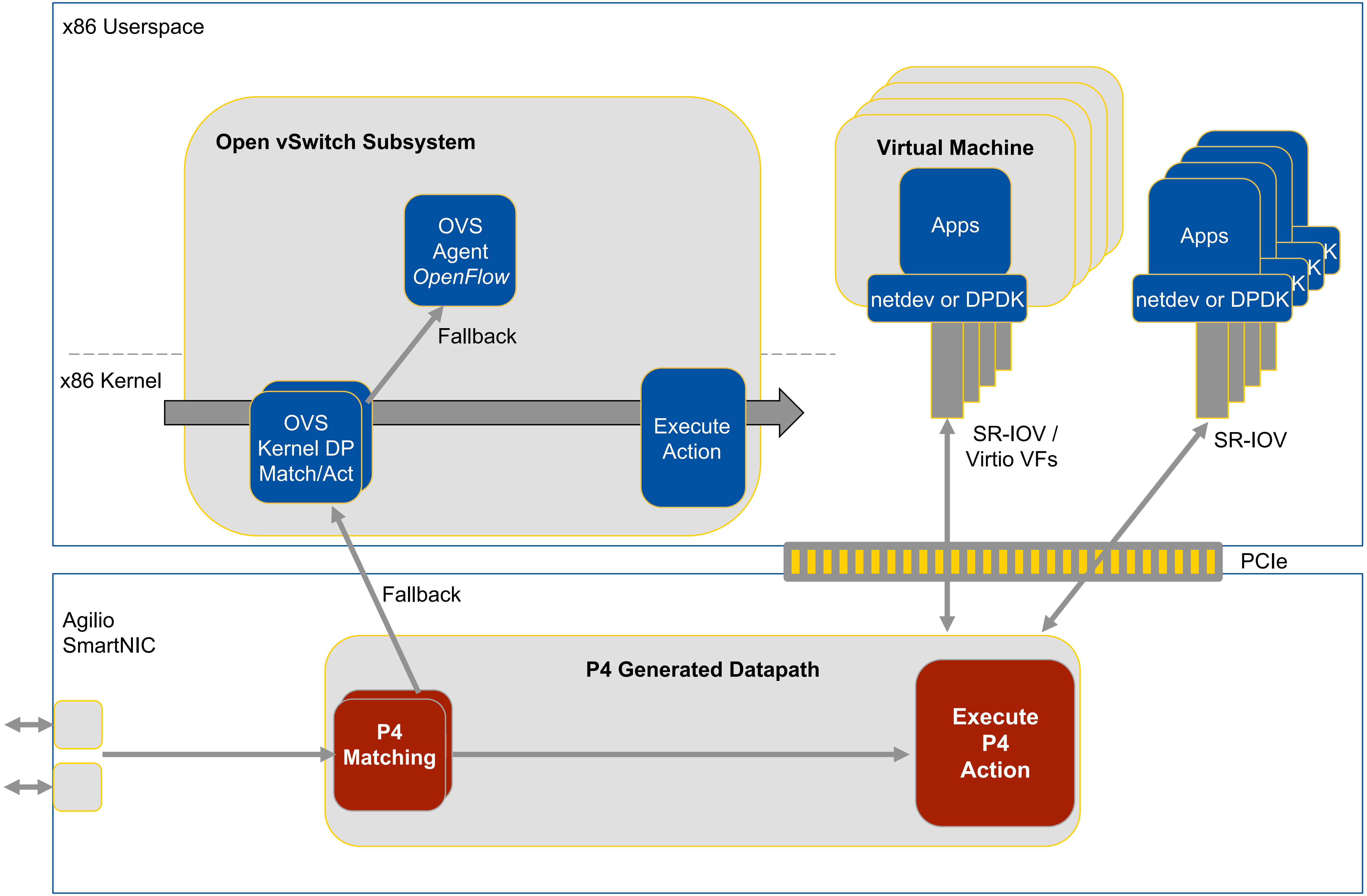
- Some flexibility
  - Easy to implement custom actions
  - Difficult to implement custom classification
  - Can implement inner protocols
- Integration effort varies
  - Can model as custom port
  - Can model as custom action



NETRONOME

P4 Datapath on SmartNIC

OVS “on” SmartNIC P4 Datapath



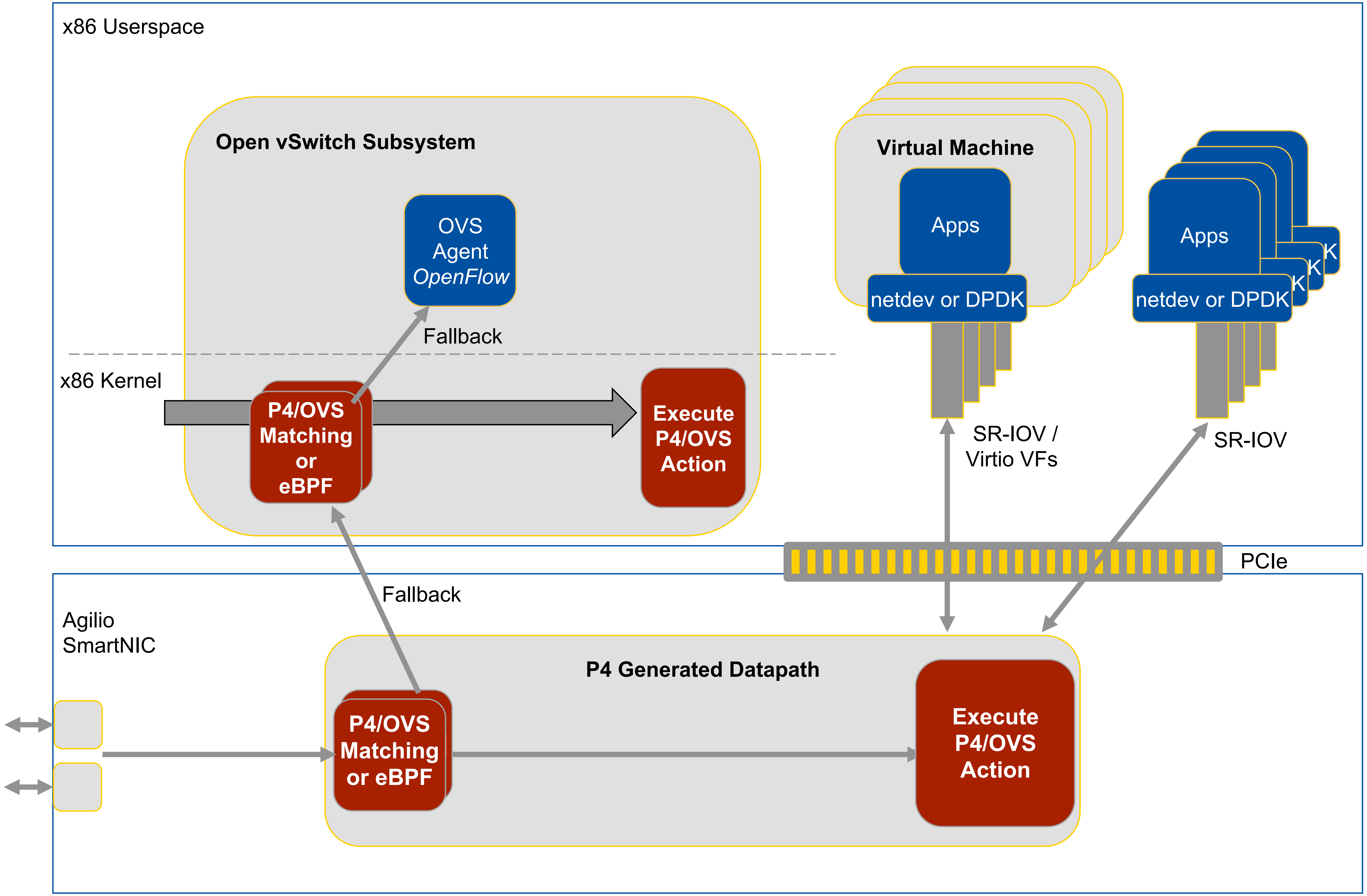
P4 Datapath on SmartNIC: Characteristics

- Some flexibility
  - In theory easy to implement offloaded behavior
  - However, OpenFlow matching is more flexible
  - Limited to what OVS on host supports
- Integration effort modest
  - Already done if offloading existing OVS code
  - Must extend OpenFlow and OVSDB or OVN if enhancing OVS

P4 Datapath In Kernel

P4 “into” OVS Datapath

NETRONOME



P4 Datapath in Kernel: Characteristics

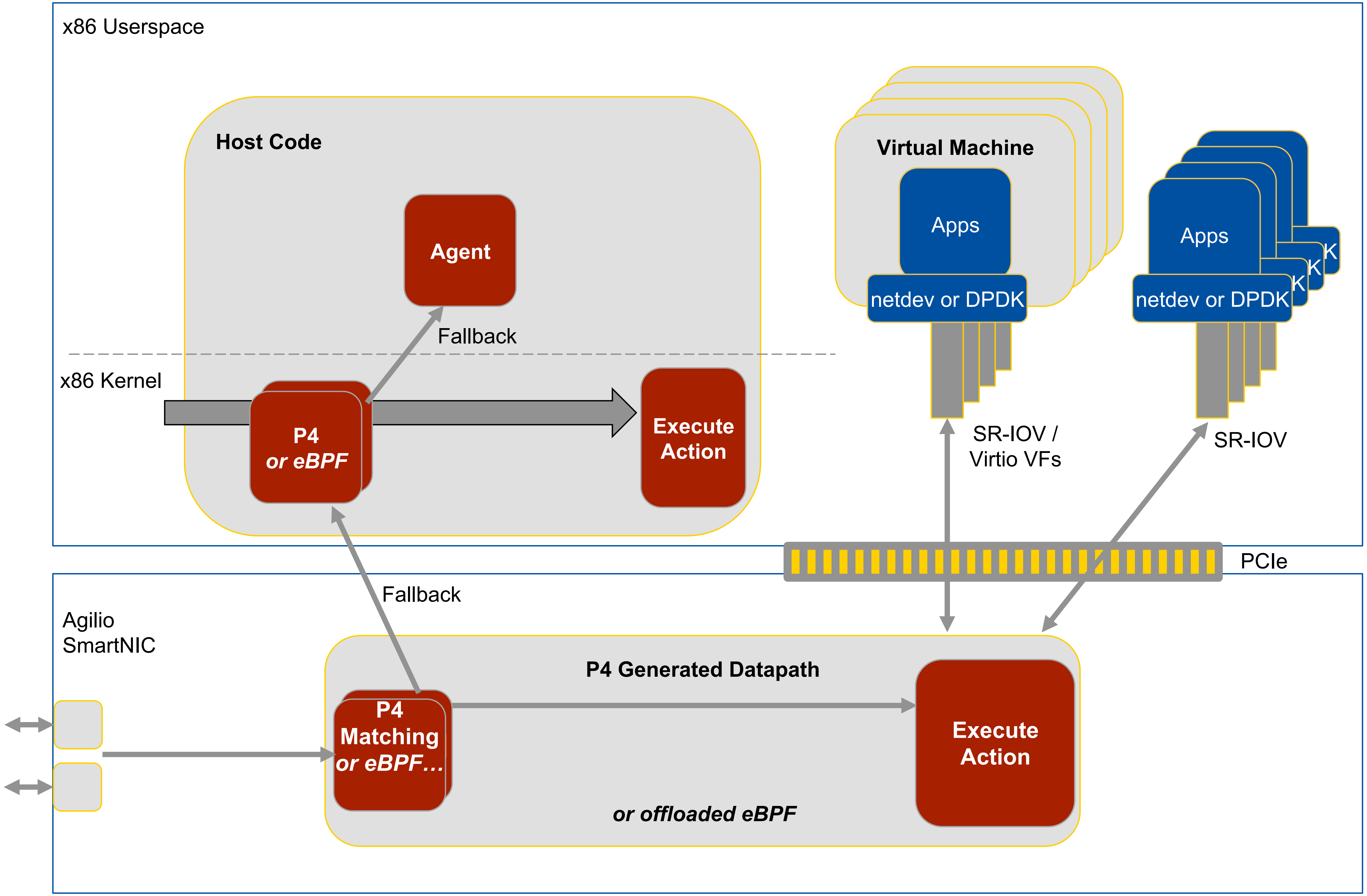
- Mixed flexibility
    - Easy to implement behavior
    - However, OpenFlow matching is more flexible
        - Regenerate program on demand
        - Implement program based on assumed model
- Integration effort considerable
    - Need to re-implement OVS on P4
    - Offloading easier once infrastructure in place

New Datapath

P4 or eBPF Datapath and Control

Open issues:

- Control Protocol - could become a callable API
- Downloading programs via OpenStack or other systems
- Scheduling VMs to run on nodes with acceleration hardware



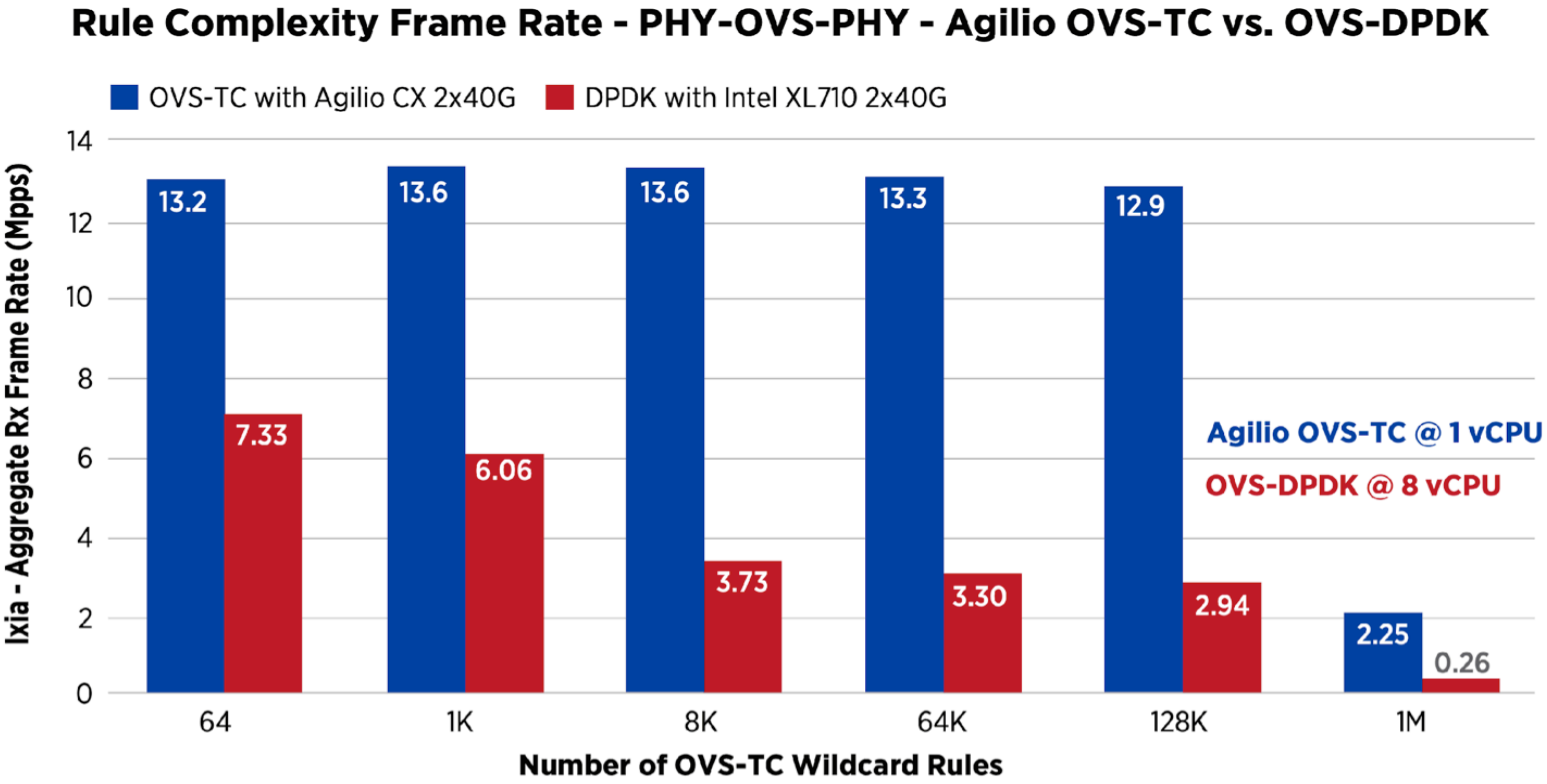
New Datapath & Control: Characteristics

- Full flexibility
  - Easy to implement behavior
  - Can deploy completely different control plane
- Integration effort considerable
  - New infrastructure required in OpenStack
    - ML2 plugin, Southbound protocol or API, etc...

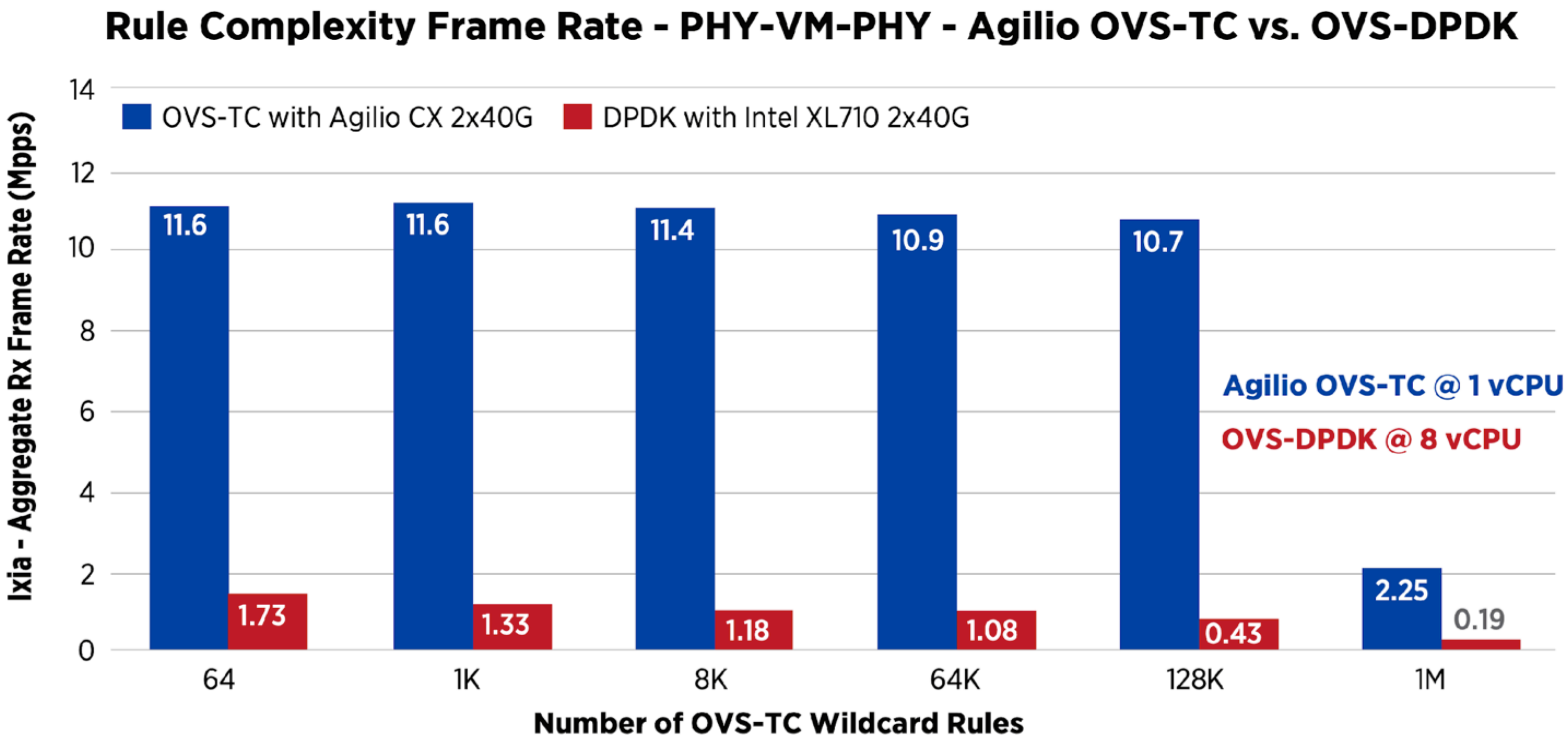
Closing Remarks

OVS Frame Rate @64k Rules+Flows

NETRONOME



OVS Frame Rate @64k Rules+Flows+VXLAN



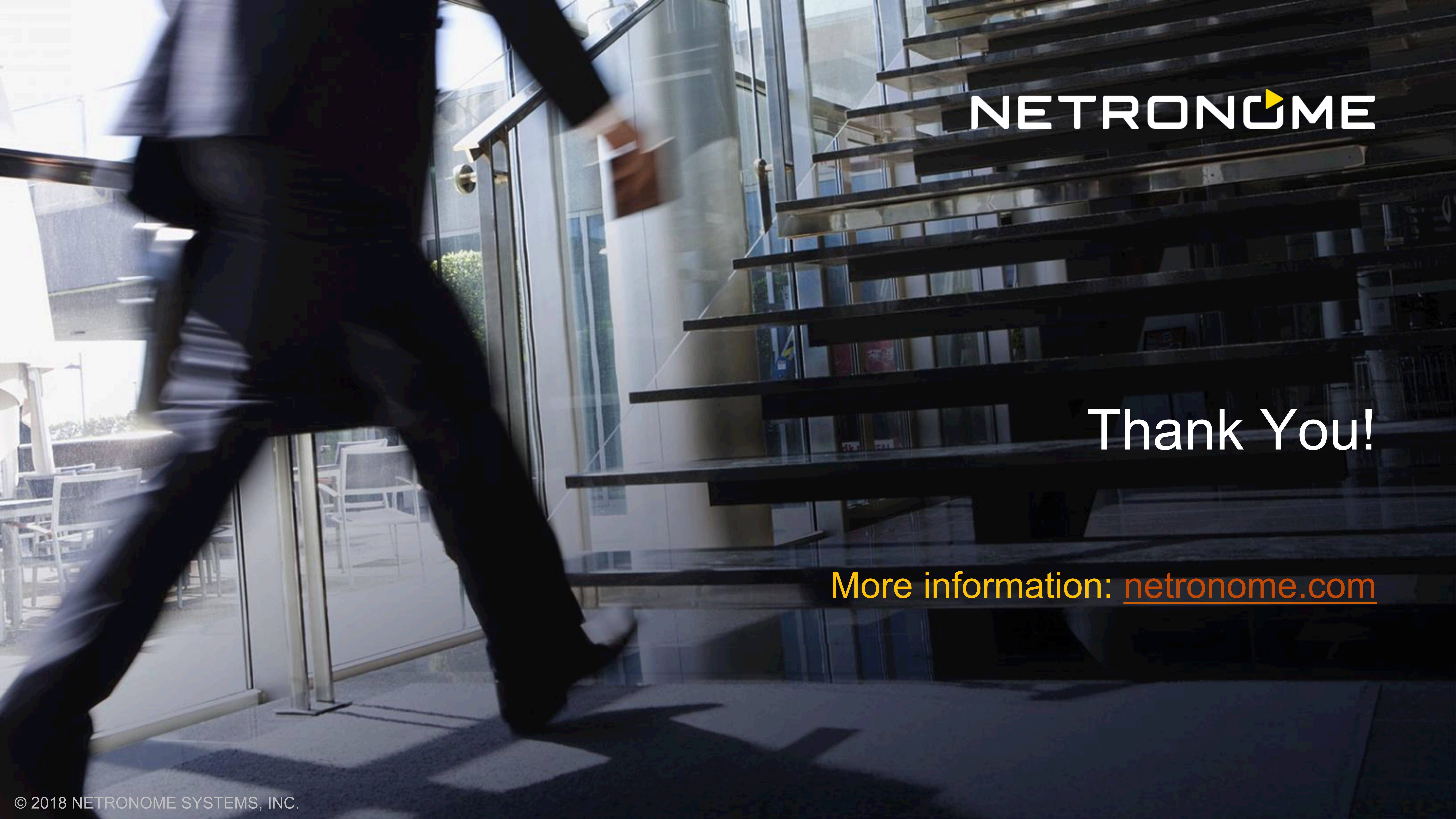
Integration / Open Sourcing Activities

Area	Activities		
Linux Drivers	Driver in upstream kernel v4.5, RHEL 7.4, Ubuntu 18.04	Representor netdevs for fallback processing and SmartNIC configuration	CoreNIC, eBPF, OVS offload feature evolution
FreeBSD Drivers	Kernel device driver implemented		
DPDK Drivers	Poll mode driver in upstream DPDK 2.2		
Open vSwitch Acceleration Integration	Present in upstream using kernel TC datapath	Feature coverage iteration in progress	
OpenStack Integration	Plugins and agents to support virtual switching acceleration present in upstream	Integration for OVS in process present in upstream	Integration for TF vRouter in process — Juniper etc...

Participation Appreciated — Join us at Linux, Open vSwitch, TF vRouter, OpenStack, p4.org, OpenSourceSDN.org

Next Steps

- Use Agilio SmartNICs with existing dataplanes
  - Use Agilio eBPF
  - Use Agilio OVS
  - Use Agilio vRouter
- Program Agilio SmartNICs
  - Use APIs (on x86 servers) - with above dataplanes
  - Program in P4 and/or C (on SmartNIC/on x86)
- Improve performance + free up server resources!



NETRONOME

Thank You!

More information: netronome.com