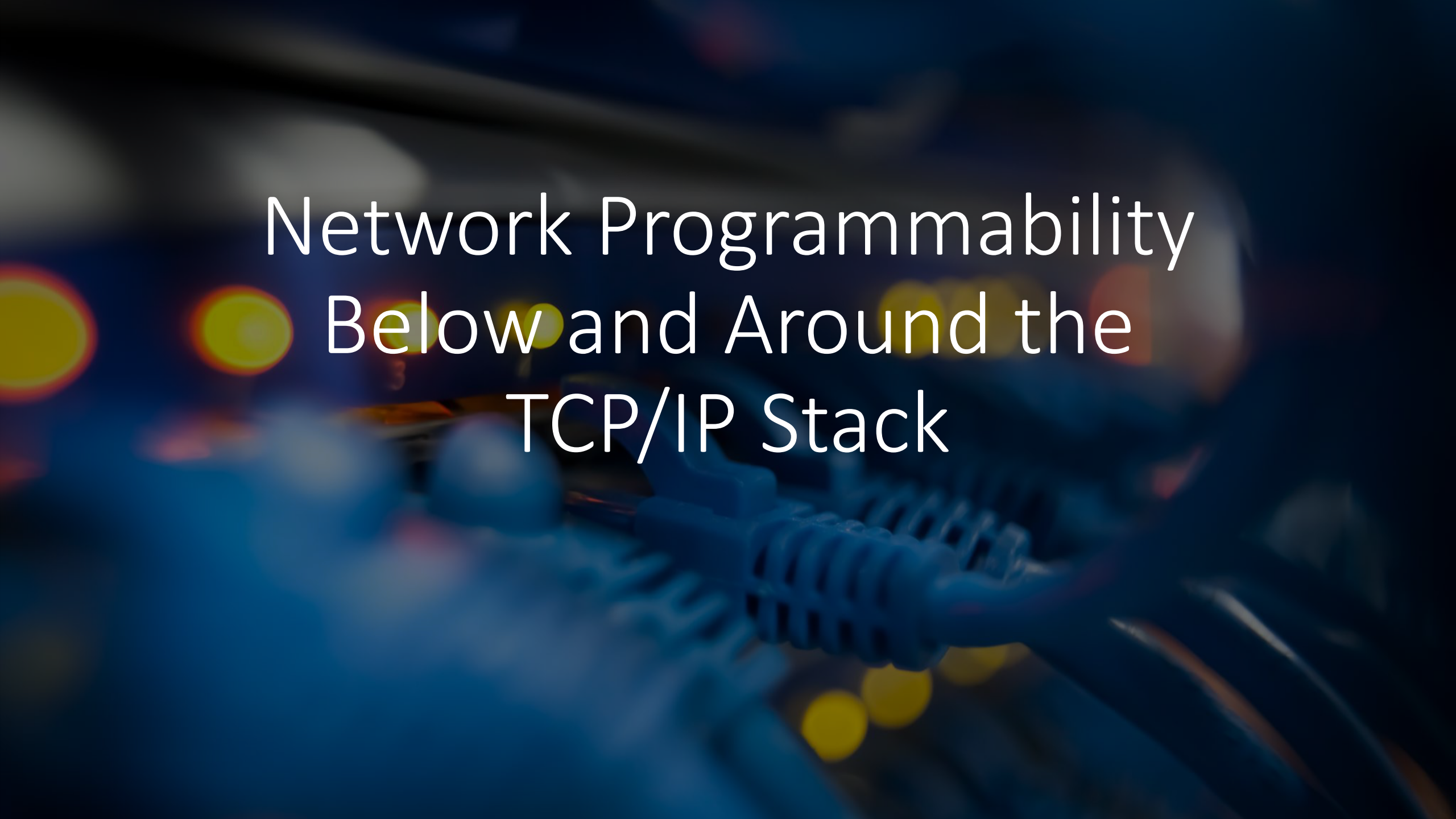
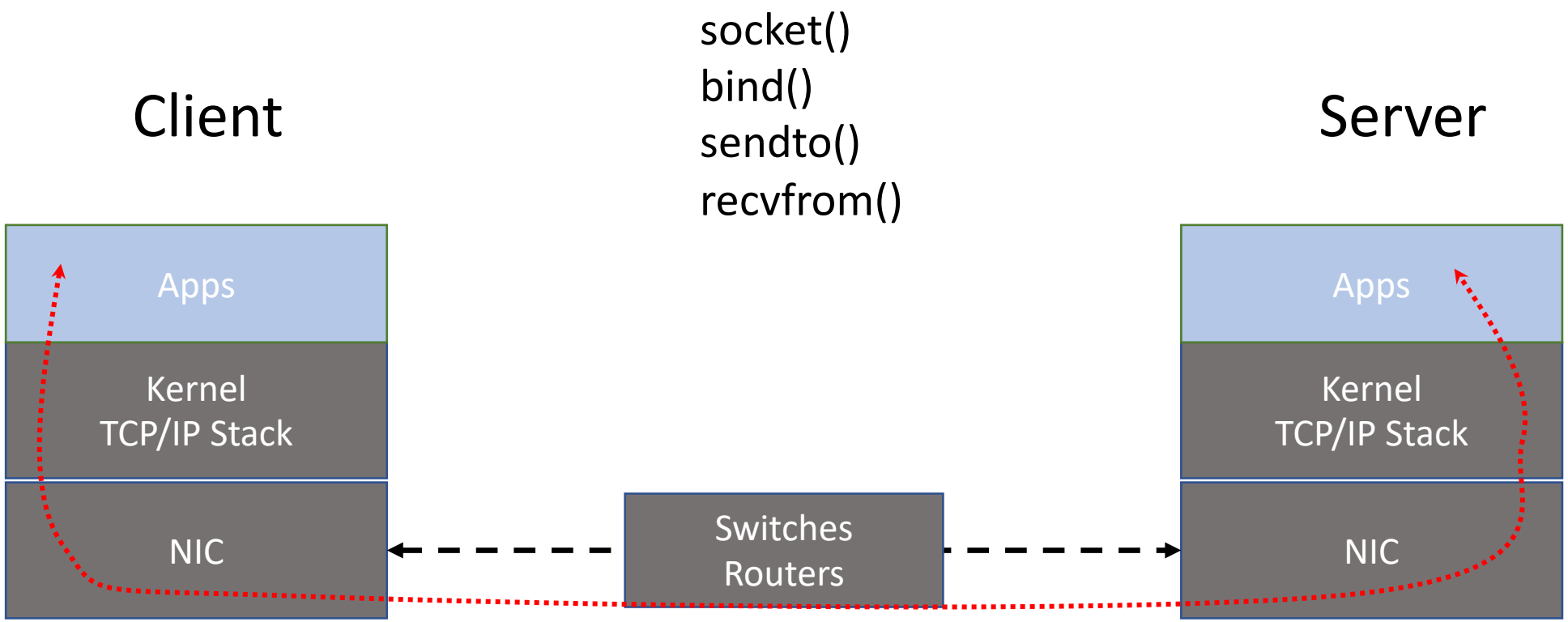


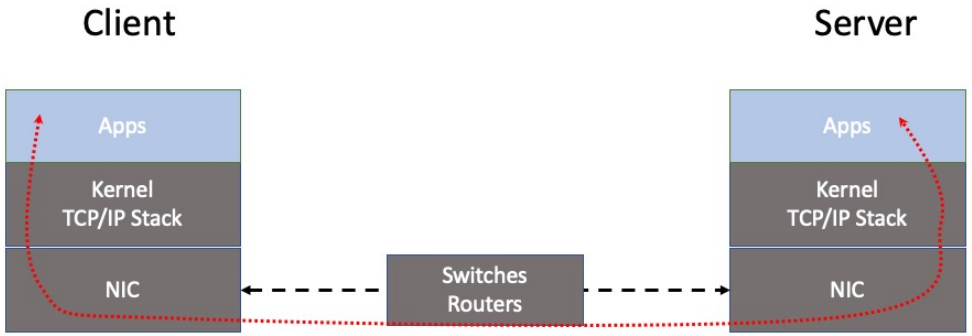
Network Programmability Below and Around the TCP/IP Stack

Network Programming since 70’



The Performance Problem

- Single threaded TCP can not saturate modern links



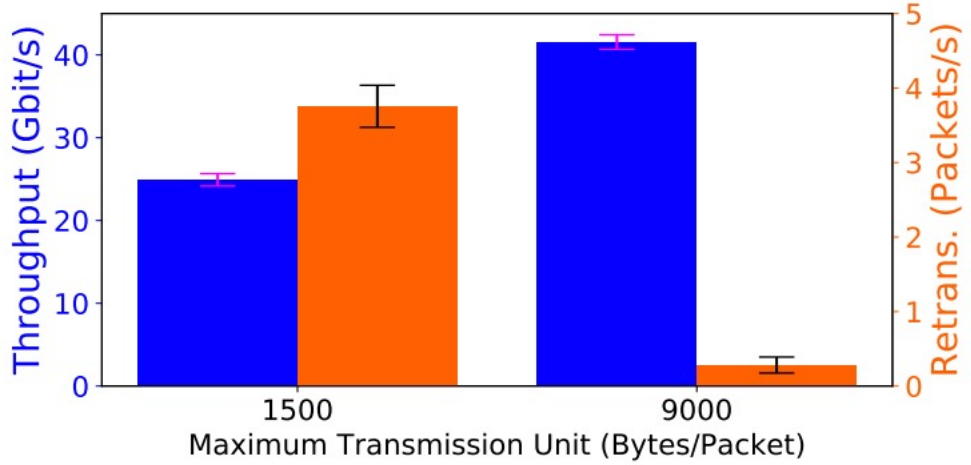
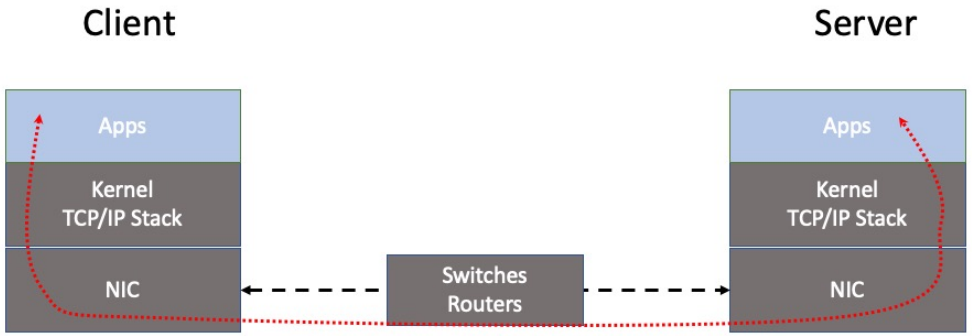


Fig. 5: Impact of the MTU on single flow performance

TCP at 100 Gbit/s – Tuning, Limitations, Congestion Control
IEEE LCN 2019

The Performance Problem

- Single threaded TCP can not saturate modern links



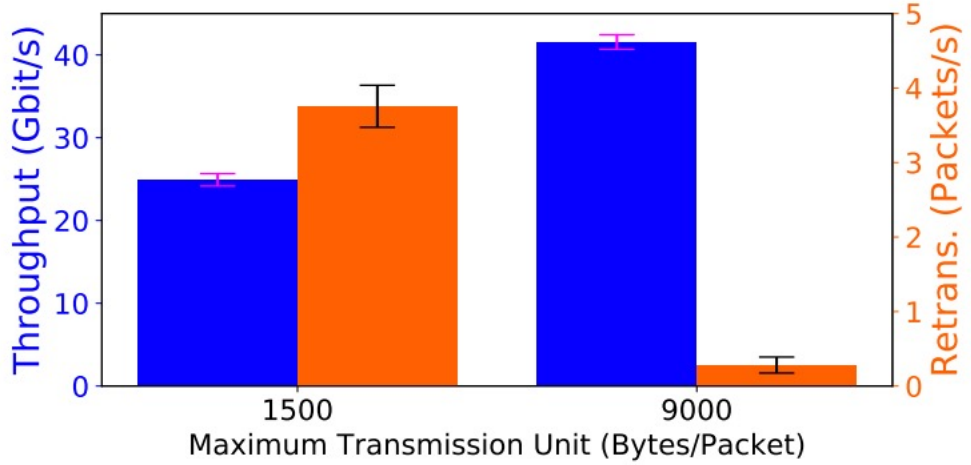


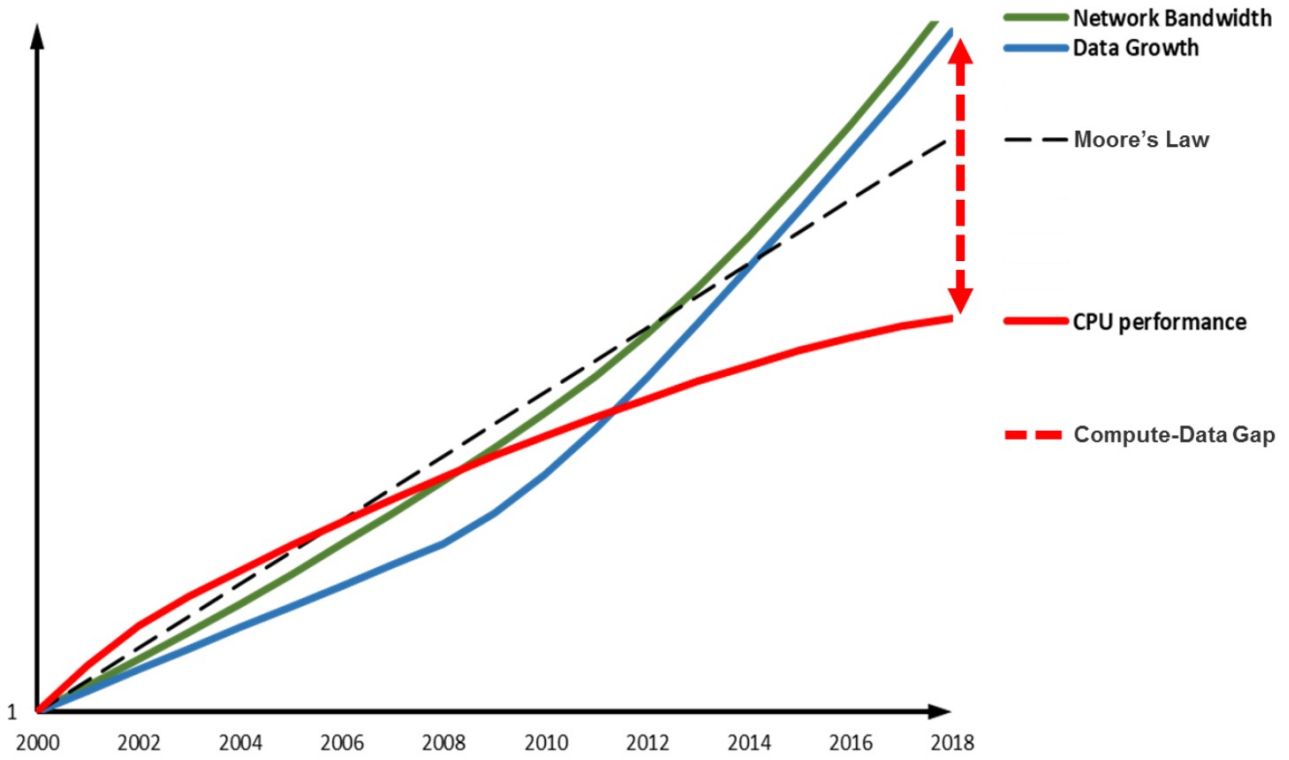
Fig. 5: Impact of the MTU on single flow performance

- Why are we only seeing this problem now?

TCP at 100 Gbit/s – Tuning, Limitations, Congestion Control
IEEE LCN 2019

Moore’s Law is Ending

- **Moore's law(1965)** is the observation that the number of transistors in a dense integrated circuit (IC) doubles about every **two years**.

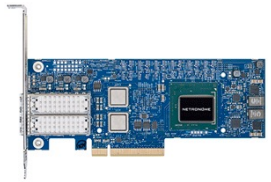


Mellanox Blog https://blog.mellanox.com/2019/08/the-end-of-moores-law-and-the-return-of-cleverness/

Moore's Law and Performance Scalability

- Fast growing demand
- Slow growing CPU performance

- We need:
  - more **efficient** codes
  - more **efficient** use of hardware
  - new hardware



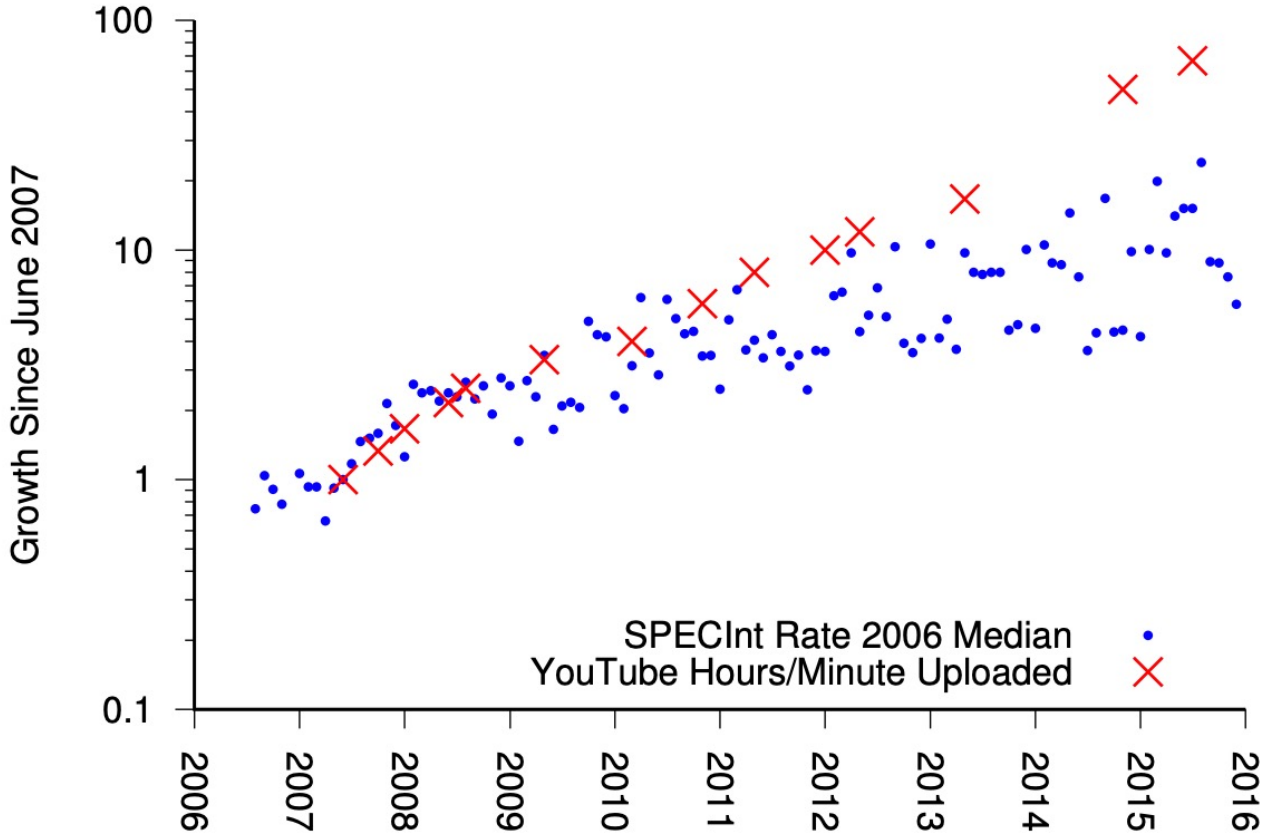
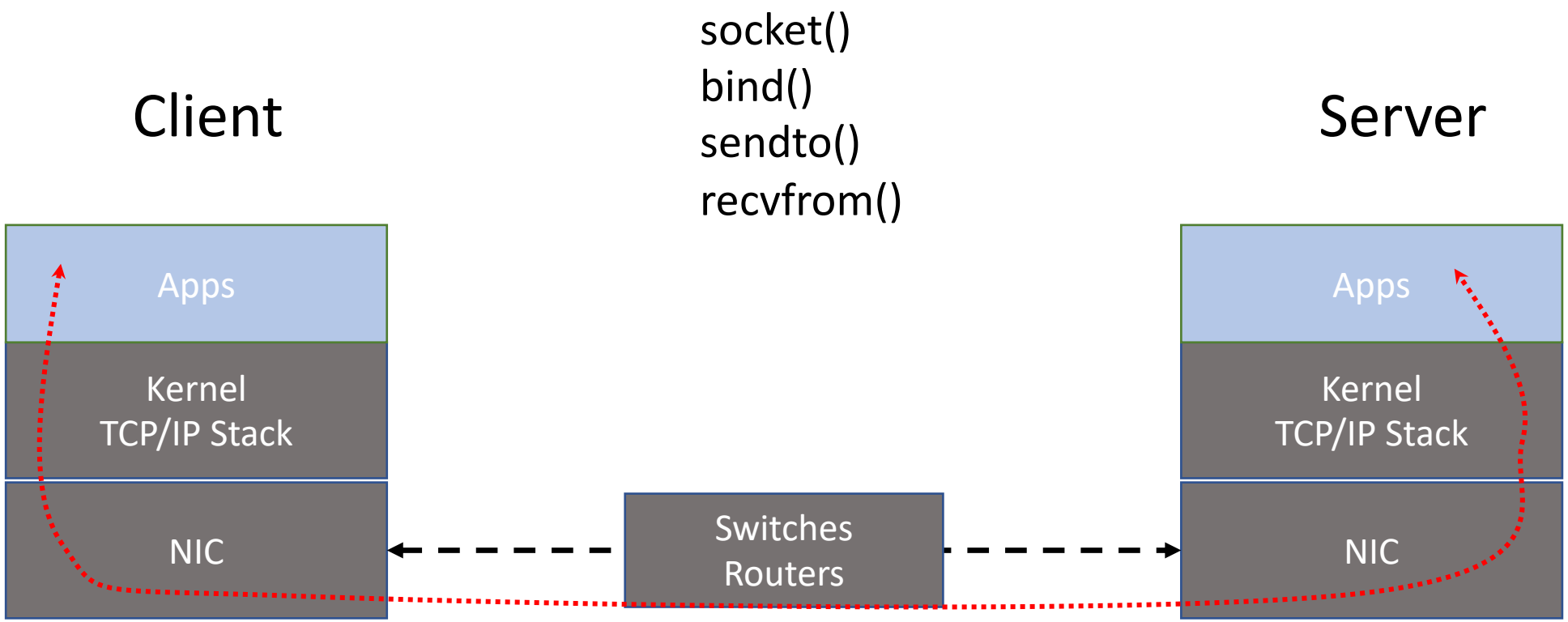
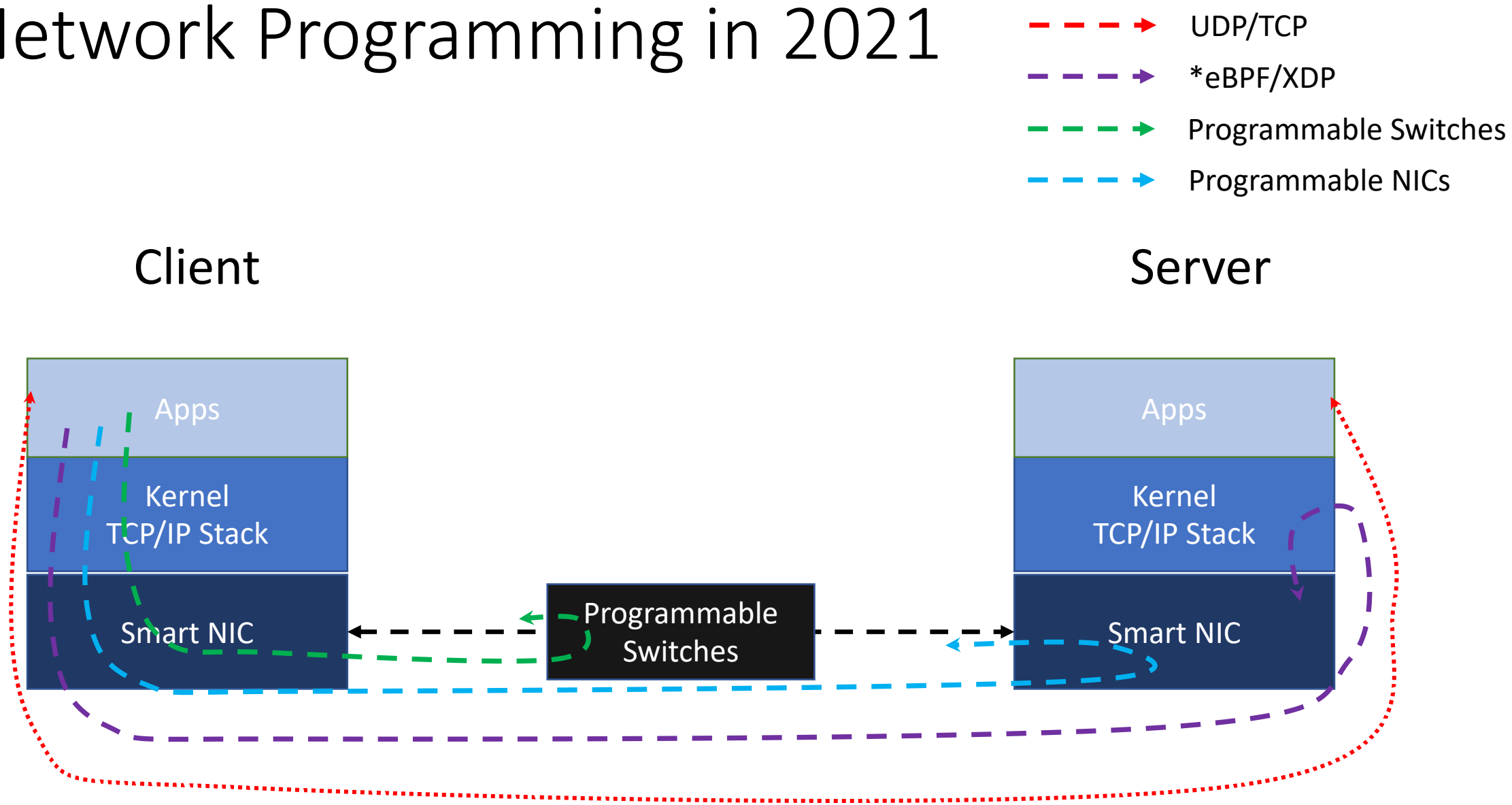


Image Source: vbech, ASPLOS '18

Network Programming since 70’

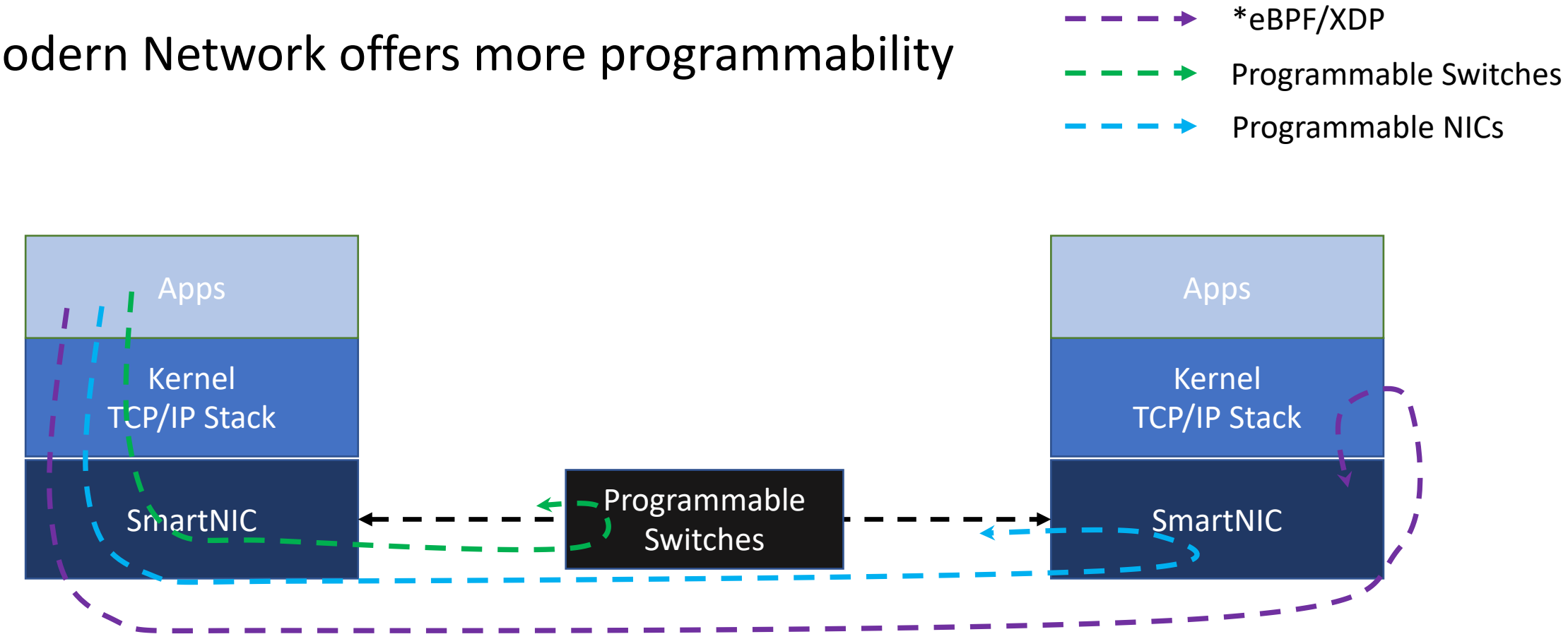


Network Programming in 2021



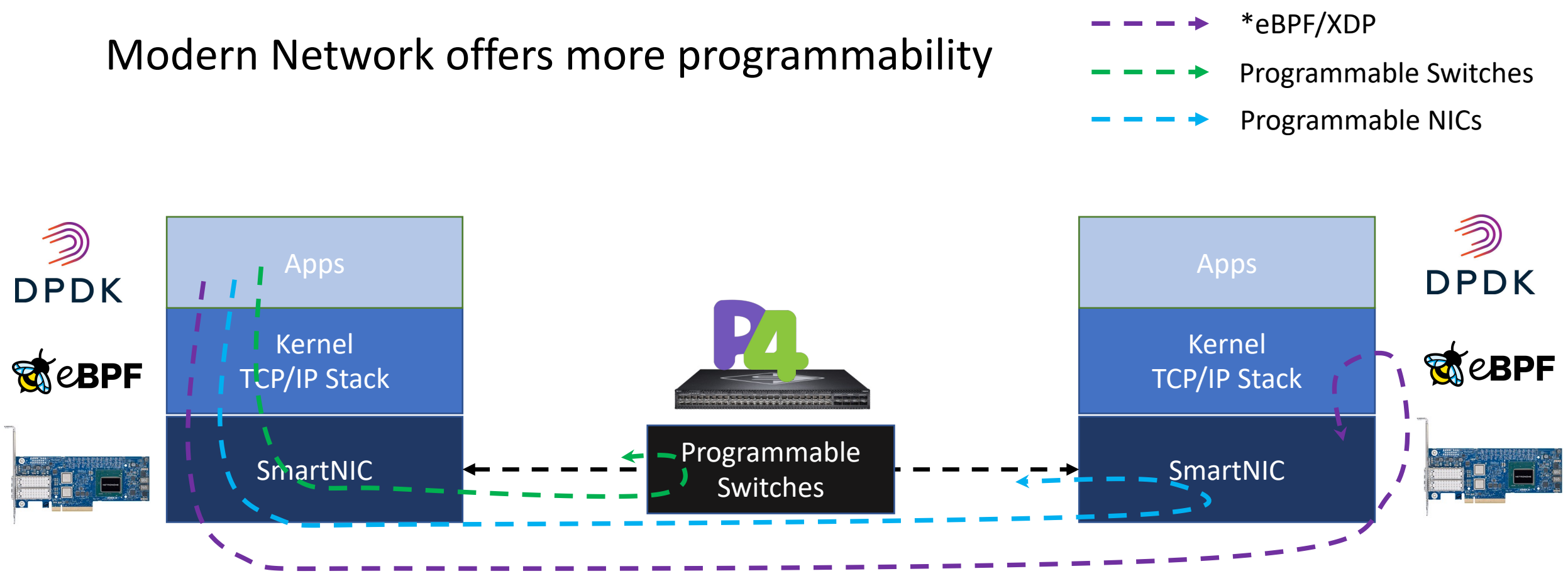
Network Programmability

Modern Network offers more programmability



Network Programmability

Modern Network offers more programmability



Network programming locations

Programmable Edge

facebook

CLOUDFLARE

Facebook: katran load balancer
Cloudflare: L4Drop Firewall

Network Stack Bypassing

DPDK

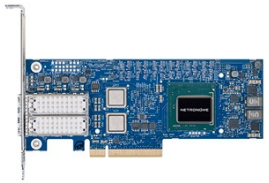
eBPF

aws

Azure

AWS Nitro SmartNIC
Azure Catapult

Smart NIC



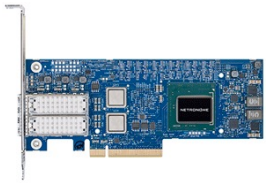
Programmable Switch

intel

Telecom Backbone,
Edge Datacenter

Programmable Switches

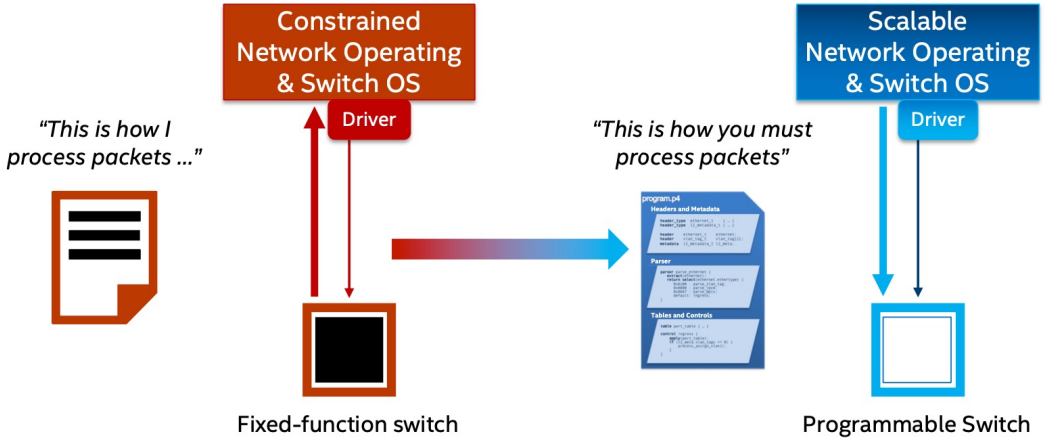
The right location for the right task




	Network Stack Bypassing	SmartNIC	Programmable Switch
Speed	<=NIC speed	10-100Gbps	3.2 Tbps
Latency	Highest	Medium	Low
State	several TB	1-4 GB	40MB

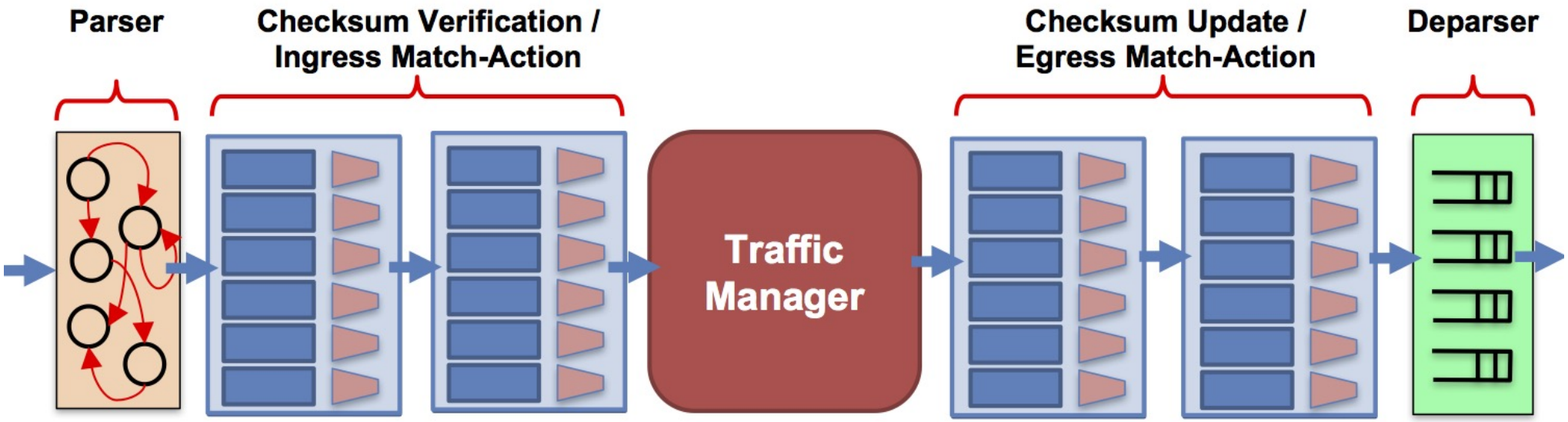
Programmable Switch

- P4: A **match-action** based programming language used to manage network dataplane
- Why P4 Programmable Switches?
  - Easy to add new features to network
  - Better packet manipulation
  - Allow in-network compute



P4: Programming Protocol-independent Packet Processors

- A open source, domain specific language for network devices.
- Defines how network packets should be processed in a pipeline manner.

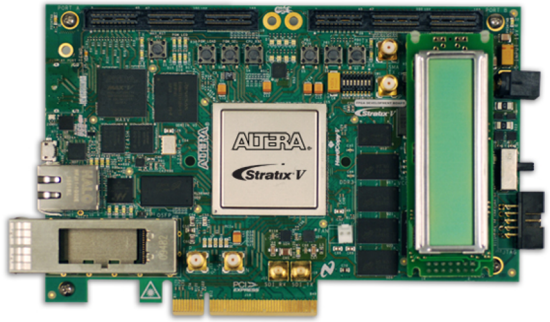


Academic Projects using Programmable Switches

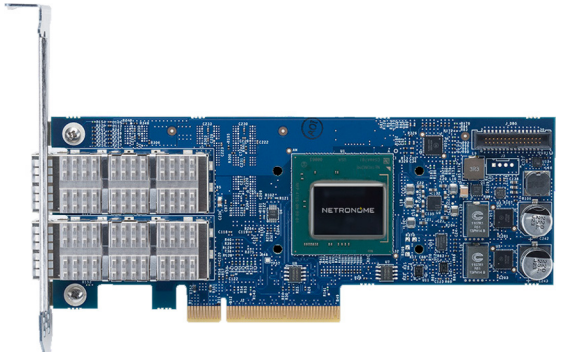
- **Netcache**(sosp17)**/Netchain**(nsdi18)**:** use P4 switch as in-network key-value store.
- ***Flow**(atc18): use P4 switch to accelerate network telemetry
- **Netpaxos**(sosr2015): total in-network simple paxos implementation
- **Pegasus**(osdi2020): use P4 switch for coherency guarantee of distributed KV

SmartNICs

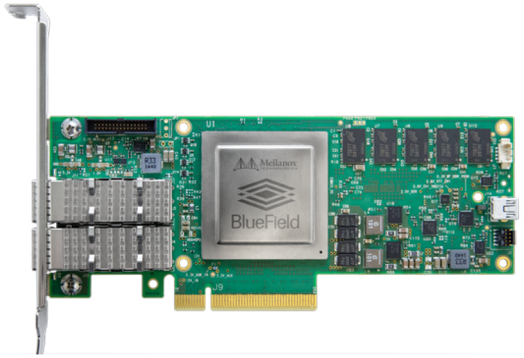
	FPGA-based SmartNICs	ASIC-based SmartNICs	SoC-based SmartNICs
Programmability	Hard	Limited	Easy
Performance	10+ cores, low latency	200+ cores, low latency	50+ cores, high latency
Development cost	High	Medium	Low



Intel
Altera



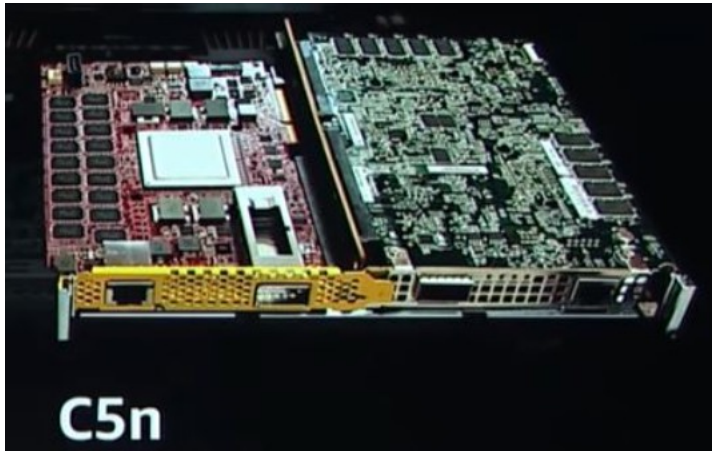
Netronome
Agilio



Mellanox
BlueField

Amazon AWS Nitro

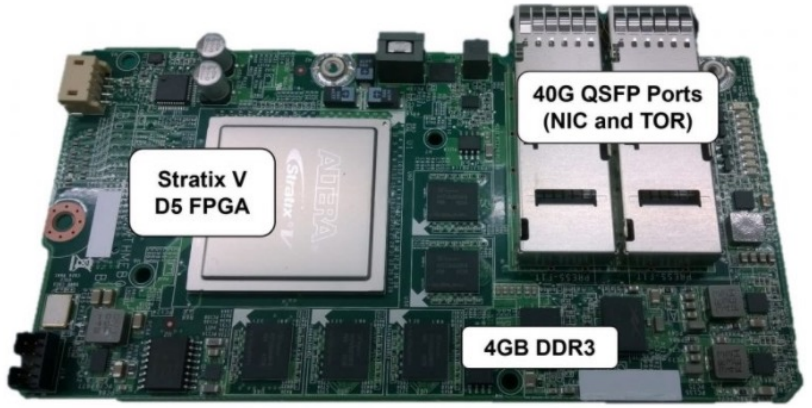
- SoC Based SmartNIC
- Offload Hypervisor functions for Network virtualization
- AWS ‘secure’ boot
- **Nitro Enclave**
- Improved Latency



Additional Reading: https://www.nextplatform.com/2020/02/03/vertical-integration-is-eating-the-datacenter-part-two/

Microsoft Azure Catapult

- FPGA Based SmartNIC
- Deployed on all Azure compute servers
- Provide Network Virtualization for compute VMs
- Can perform other tasks:
  - Crypto, QoS, Storage Acceleration



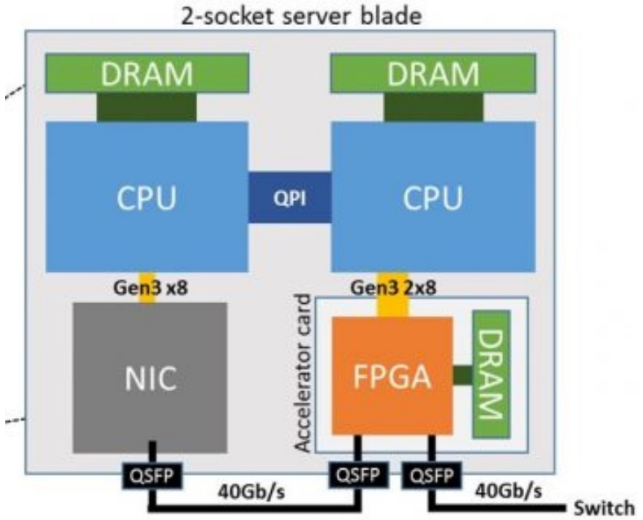


Image sorce: https://www.microsoft.com/en-us/research/publication/configurable-cloud-acceleration/

Additional Reading: https://www.nextplatform.com/2020/02/03/vertical-integration-is-eating-the-datacenter-part-two/

Nvidia SmartNIC + GPU

ANNOUNCING
NVIDIA BLUEFIELD-2X
AI to Data Center Infrastructure

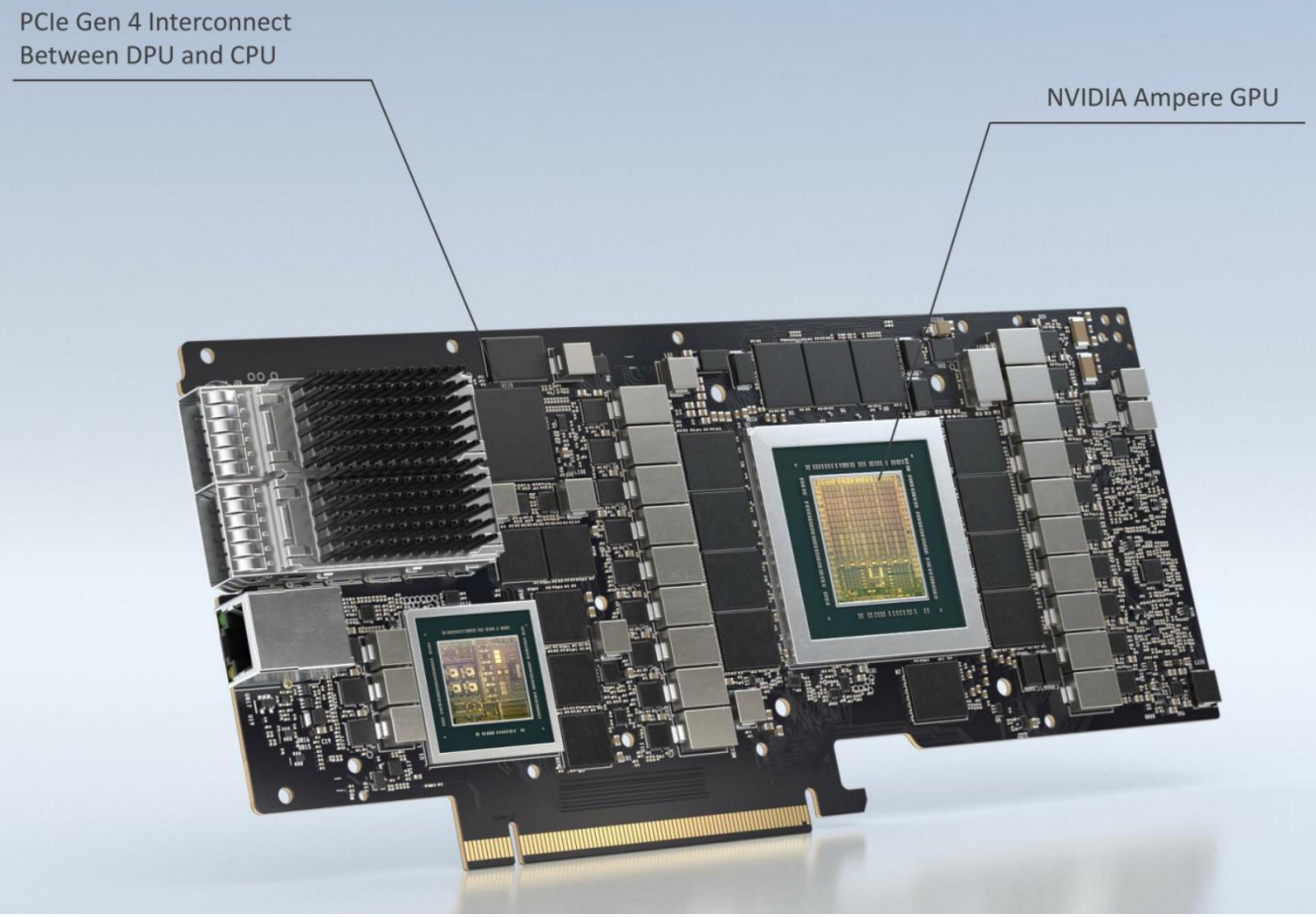
Anomaly Detection & Automated Response

Real-Time Traffic Analytics at Line Rate

Host Introspection to Identify Malicious Activity

Dynamic Security Orchestration

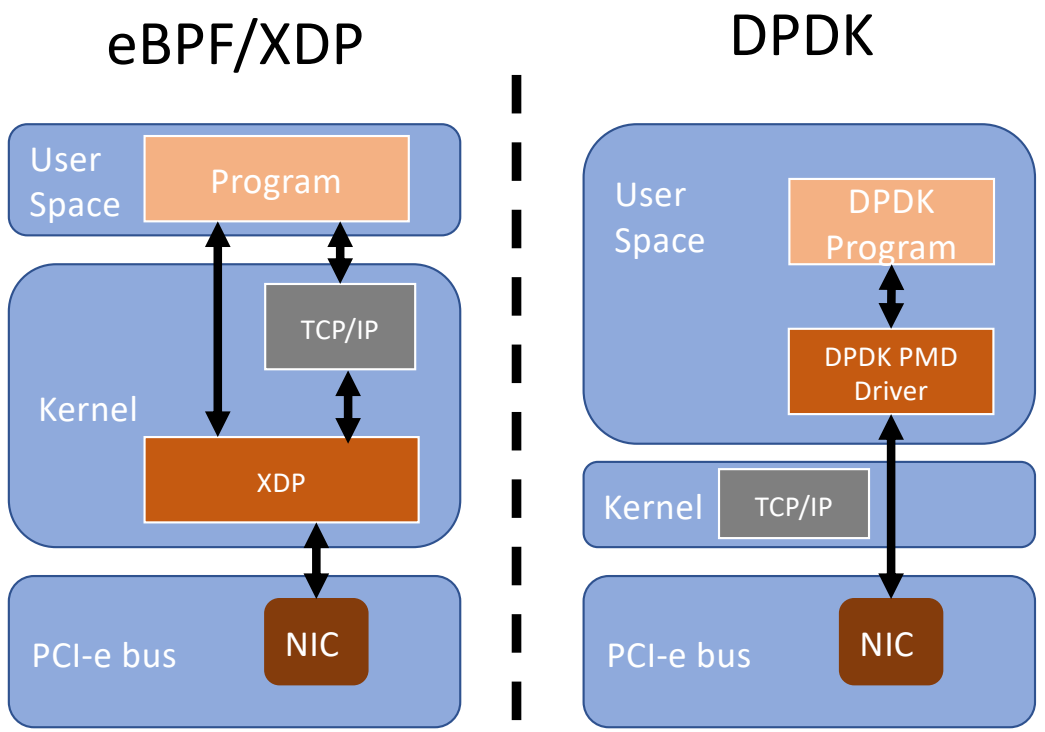
Online Analytics of Uploaded Video



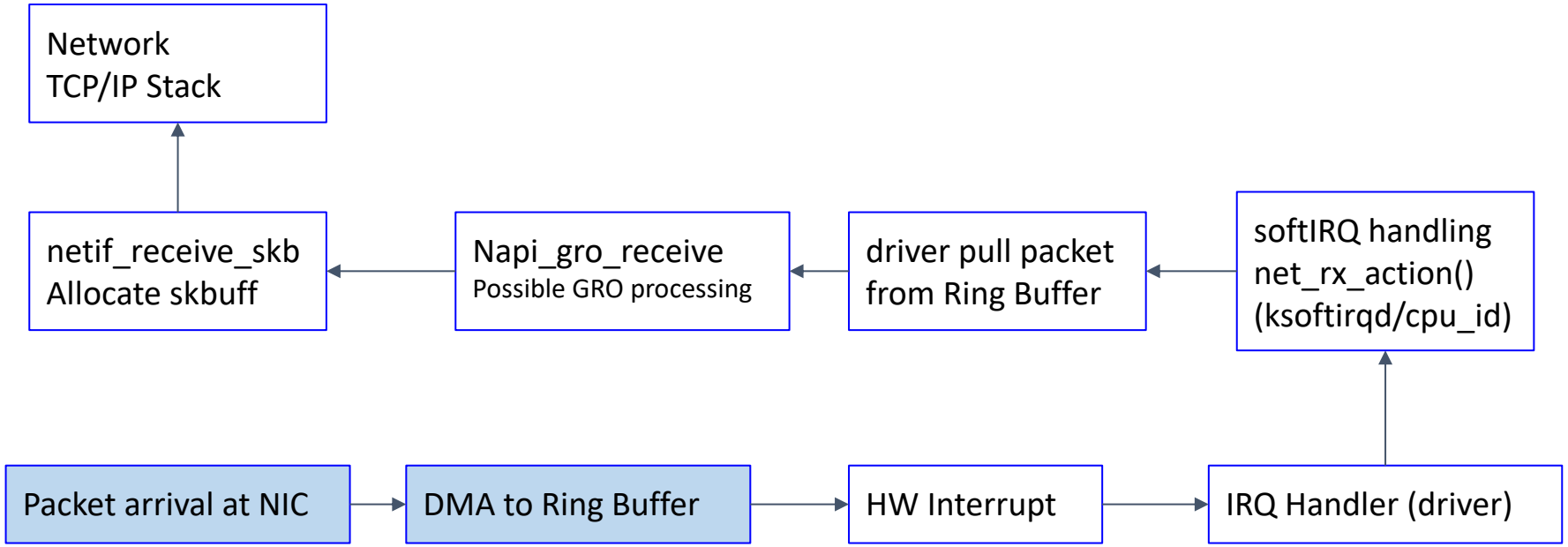
Additional Reading: https://www.nextplatform.com/2020/02/03/vertical-integration-is-eating-the-datacenter-part-two/

Network Stack Bypassing

- Extended Berkeley Packet Filter
  - Push packet processing program into Kernel
- Data Plane Development Kit
  - move packets directly into user space.
- Two philosophies, same goal



Linux Kernel Packet Path

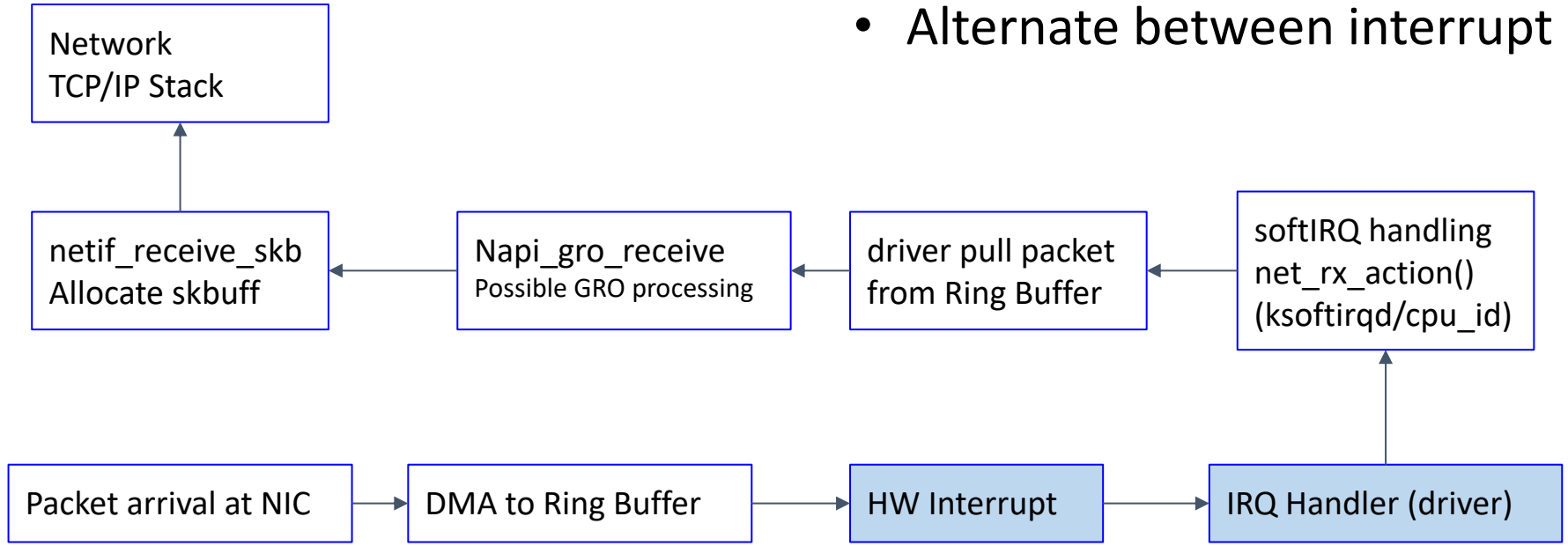


More Kernel details:
https://blog.packagecloud.io/eng/2016/06/22/monitoring-tuning-linux-networking-stack-receiving-data/

Linux Kernel Packet Path

New API Interrupt mitigation:

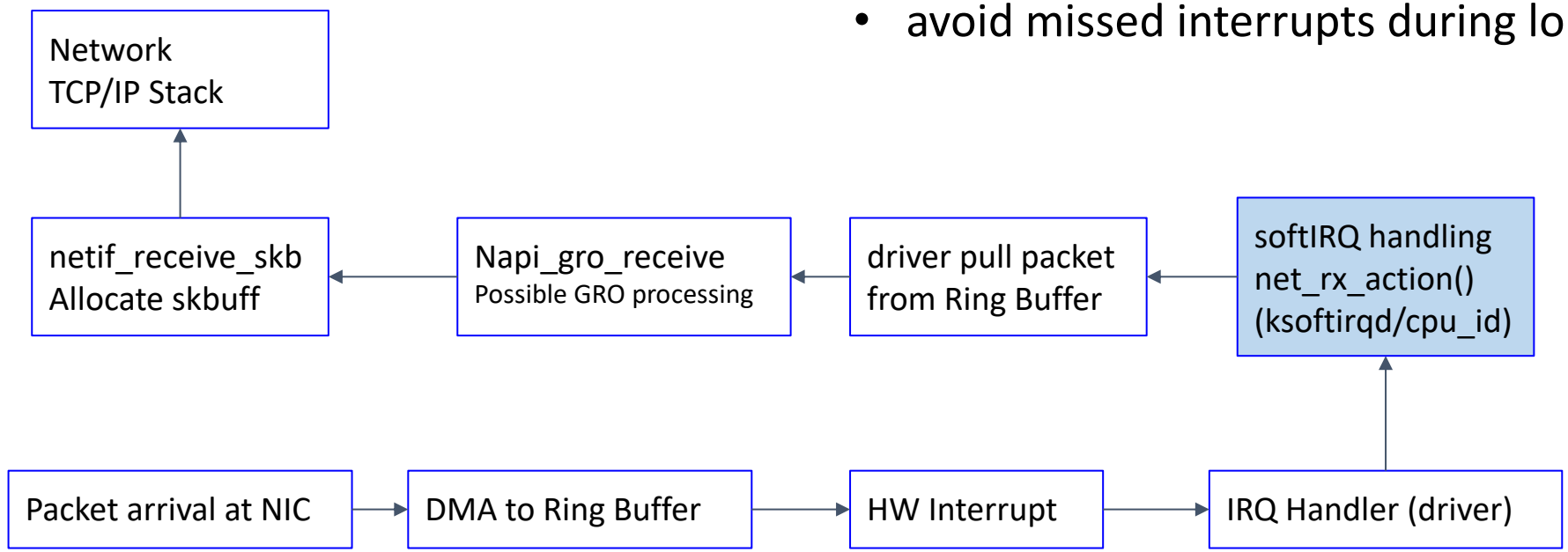
- Avoid too many HW interrupts
- Alternate between interrupt and poll modes.



Linux Kernel Packet Path

SoftIRQ:

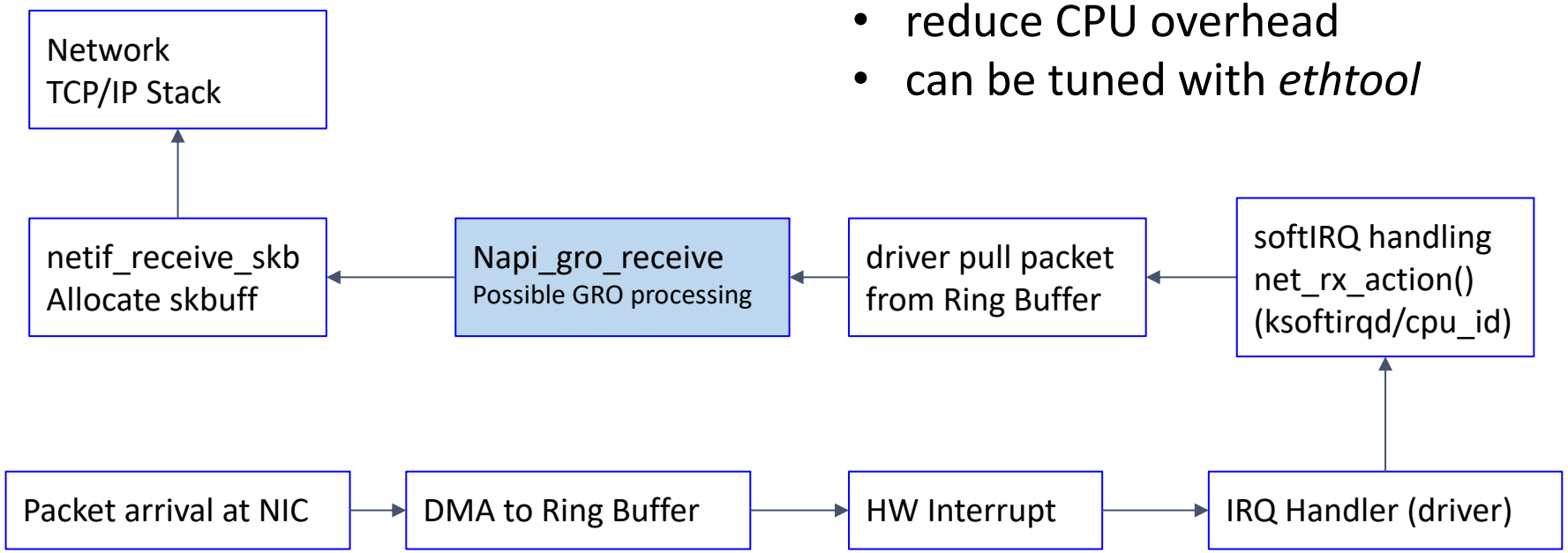
- execute code outside context of interrupt handler
- avoid missed interrupts during long running actions.



Linux Kernel Packet Path

Generic Receive Offloading **GRO**(LRO):

- group multiple packets into single buffer
- reduce CPU overhead
- can be tuned with *ethtool*



Linux Kernel Packet Path

Socket Buffer is the meta data used for packet processing in network stack

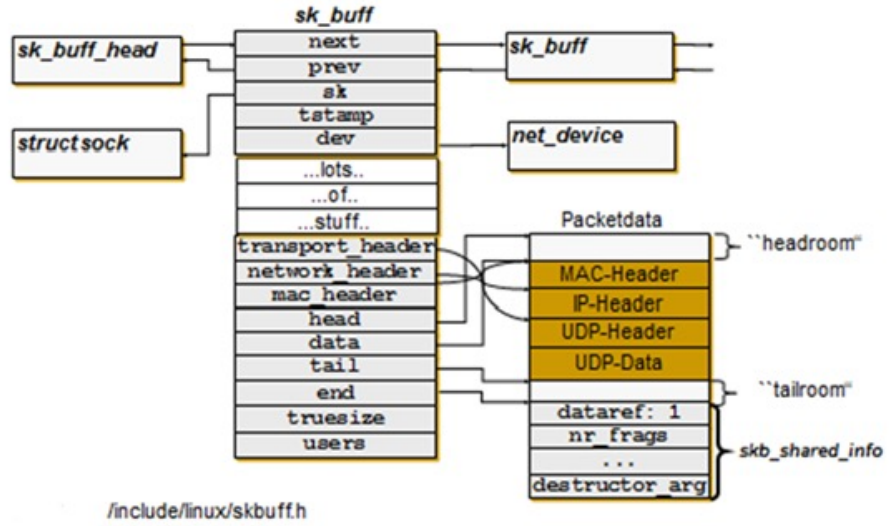
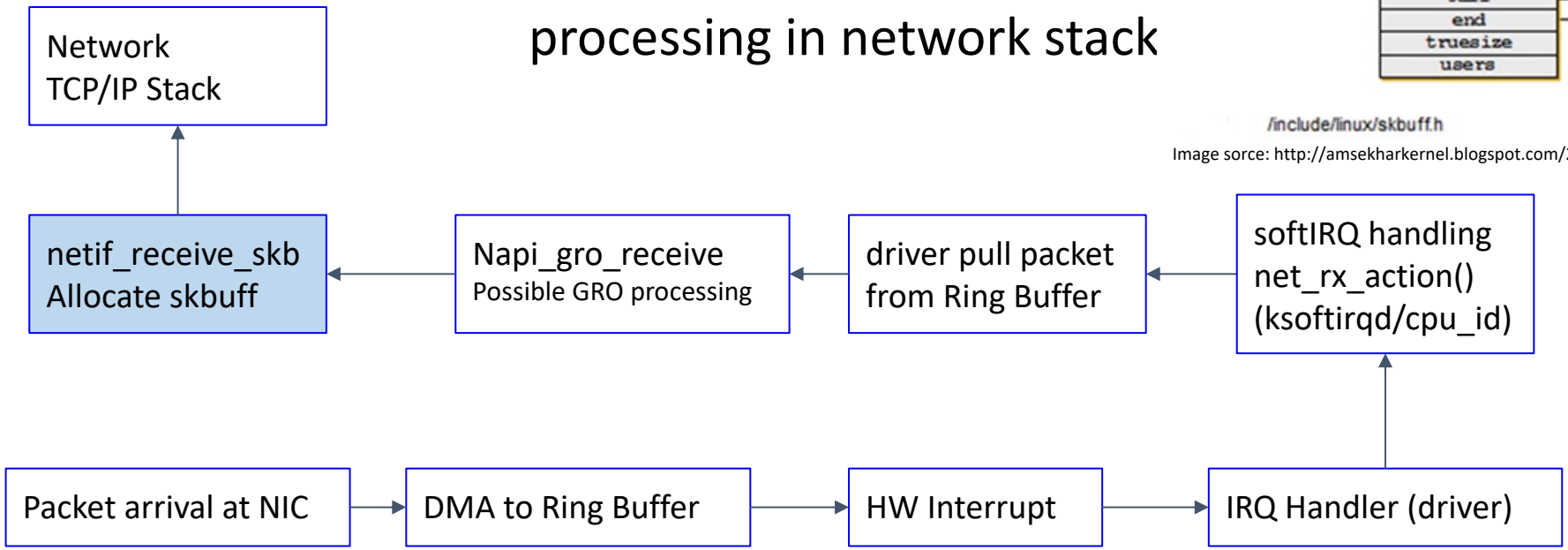
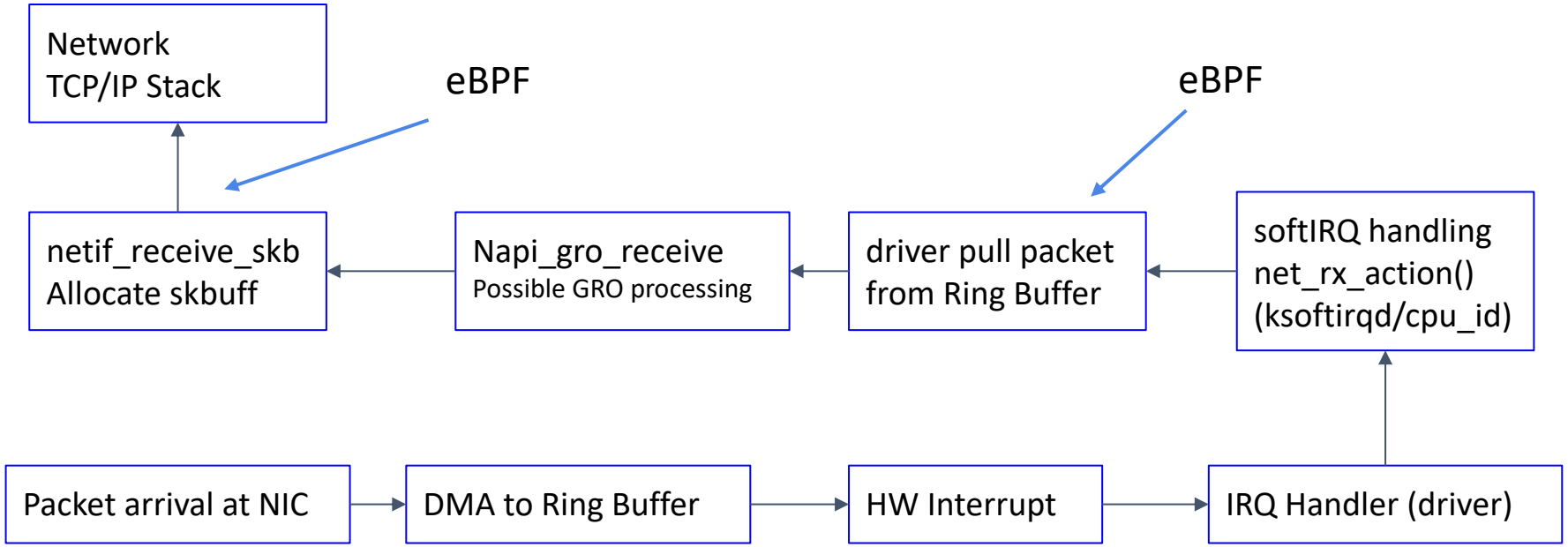


Image sorce: http://amsekharkernel.blogspot.com/2014/08/what-is-skb-in-linux-kernel-what-are.html



eBPF: in-kernel VM Below Network Stack



BPF -> eBPF -> XDP

- Berkeley Packet Filter(1993)
  - Register based VM that runs in kernel
  - Originally designed to process packets tcpdump/libpcap
  - BPF programs are Bytecode programs and JITted during runtime.

```
$ sudo tcpdump -p -ni eth0 -d "ip and udp"
(000) ldh      [12]
(001) jeq      #0x800           jt 2    jf 5
(002) ldb      [23]
(003) jeq      #0x11            jt 4    jf 5
(004) ret      #65535
(005) ret      #0
```

Load a half-word (2 bytes) from the packet at offset 12.

Check if the value is 0x0800 (IP packet)

Load byte from a packet at offset 23 (IP protocol field)

Check if the value is 0x11 (UDP)

Return success

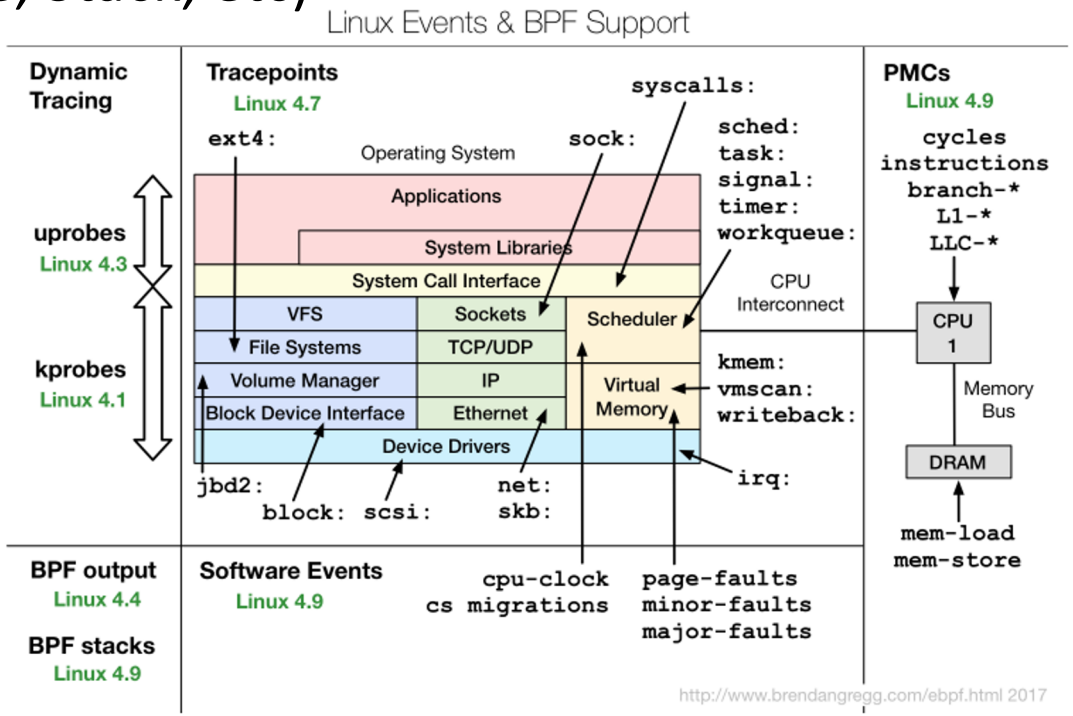
Return fail

McCanne, S. and Jacobson, V., 1993, January. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *USENIX winter* (Vol. 46).

https://blog.cloudflare.com/bpf-the-forgotten-bytecode/

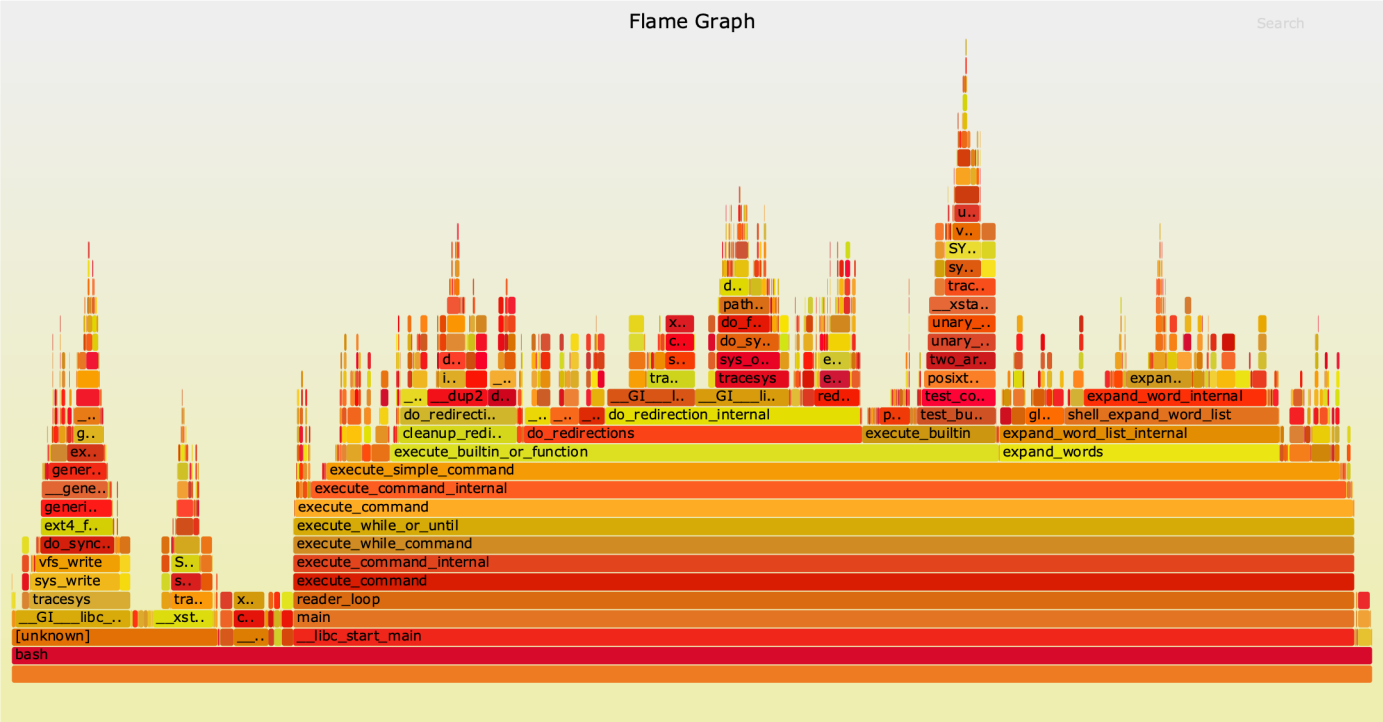
BPF -> eBPF -> XDP

- extended BPF(2014)
  - Upgrade from 2 32-bit registers to 10 64-bit registers
  - Add new BPF kernel hookup locations: kprobes/ tracepoints, etc
  - Support data structures(Hash, Array, Queue, Stack, etc)
  - Access kernel helper functions



BPF -> eBPF -> XDP

- Can be used other than packet processing
- Performance profiling with FlameGraph
  - https://github.com/brendangregg/FlameGraph

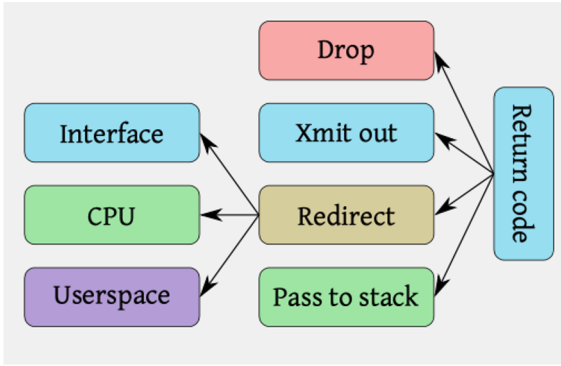


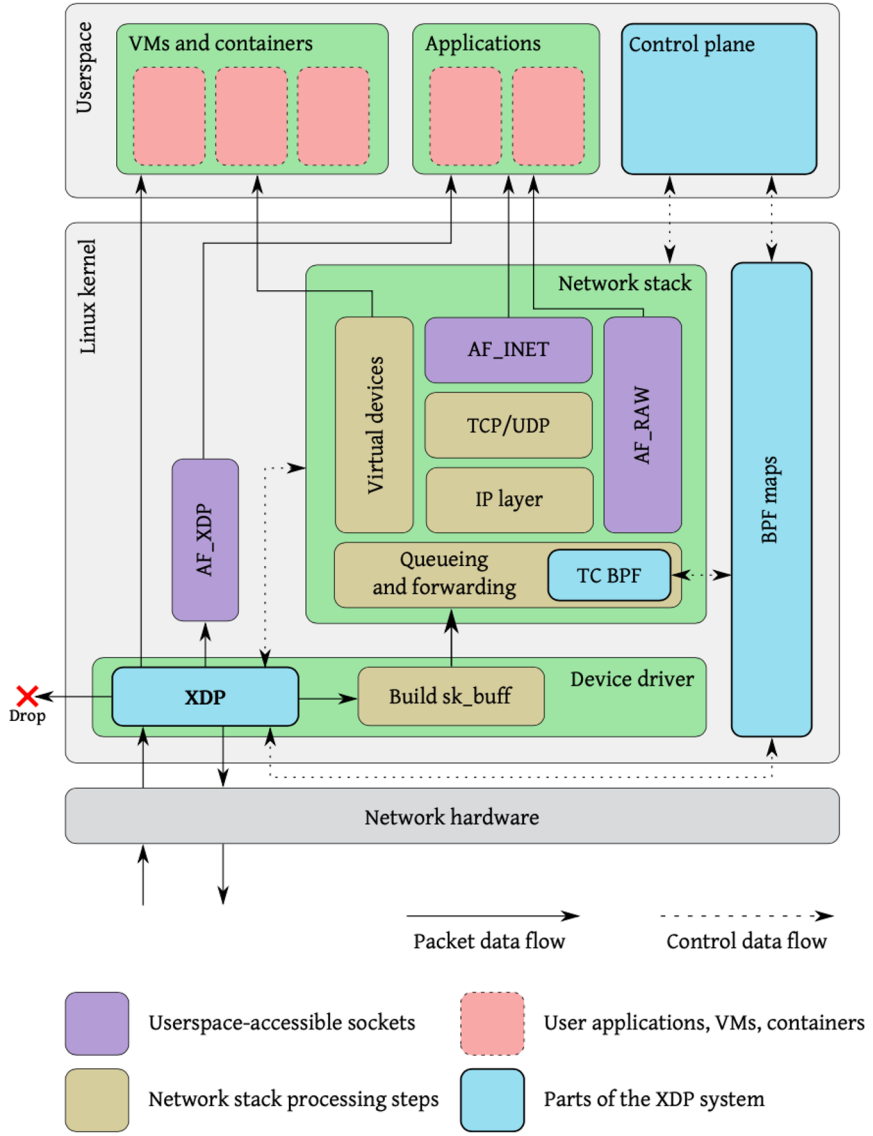
BPF -> eBPF -> XDP

- XDP(eXpress Data Path)(2016)
  - Lowest possible location to run eBPF in packet path
  - Performance(mlx4 40Gb/s With DDIO)
    - Packet drop 20M pps per CPU(touch data)
    - Forwarding 14M pps per CPU
    - Provides multiple return codes
- XDP v.s. DPDK
  - Different philosophy, run code in Kernel v.s. Userspace
  - Poll mode XDP can work with DPDK too!
- Current heavy users of XDP
  - Cloudflare: DDoS mitigation (first killer app for XDP)
  - Cilium: Networking and security for containers
  - Facebook: katran load balancer etc.

BPF -> eBPF -> XDP

- XDP(eXpress Data Path)(2016)
  - Lowest possible location to run eBPF in packet path
  - Performance(mlx4 40Gb/s With DDIO)
    - Packet drop 20M pps per CPU(touch data)
    - Forwarding 14M pps per CPU
    - Provides multiple return codes



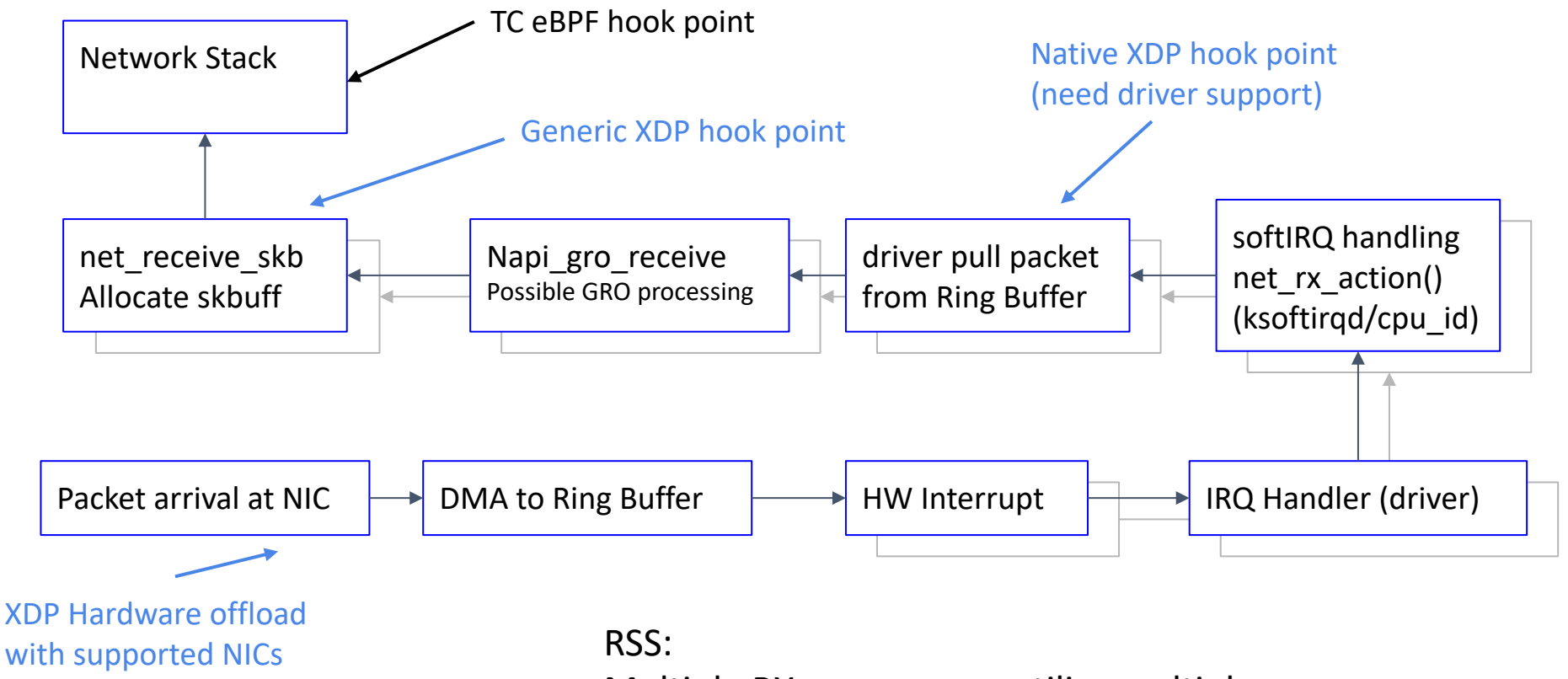


The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel, CoNEXT'18

BPF -> eBPF -> XDP

Lots of potentials

- Various running locations:



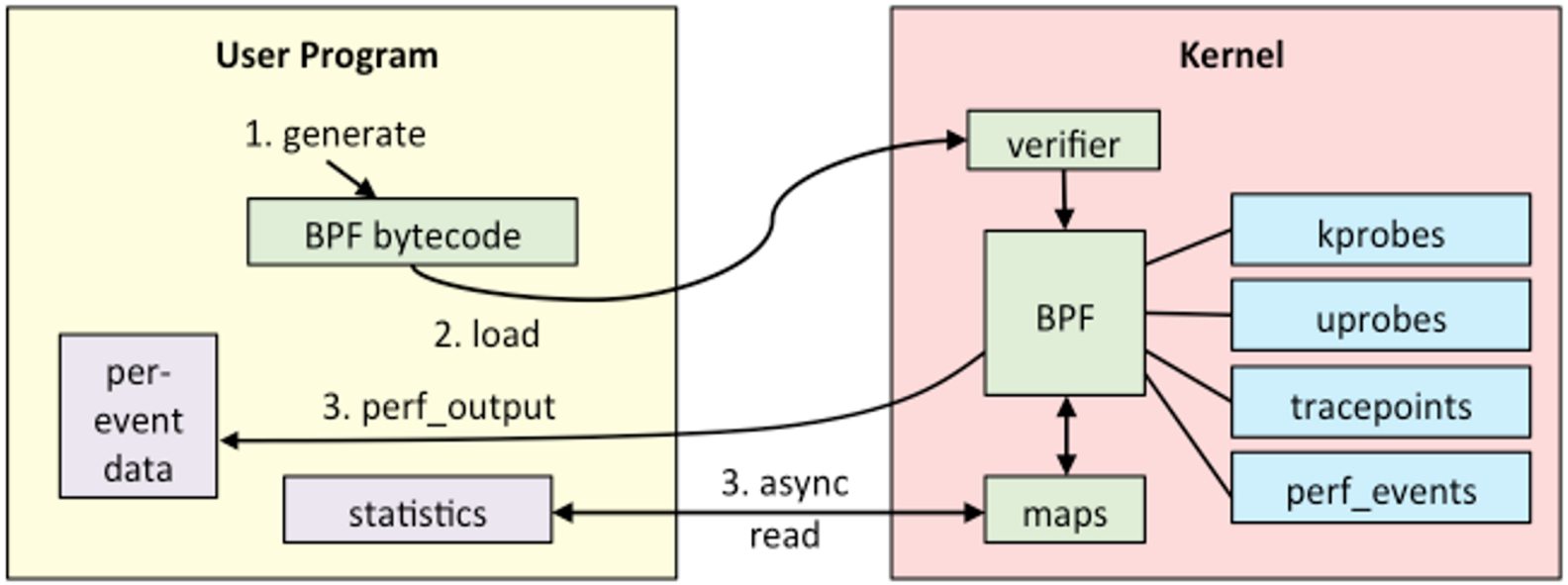
RSS:
Multiple RX_queues can utilize multiple
CPUs for parallel processing load balancing

BPF(1993) -> eBPF(2014) -> XDP(2016)

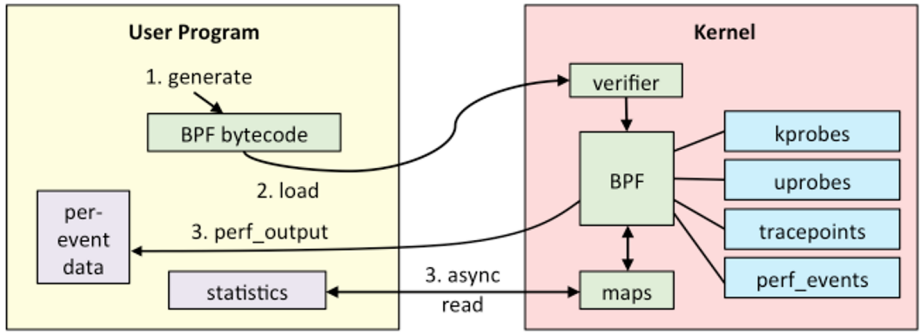
- eBPF/XDP is evolving very fast
  - New features with almost every kernel update
    - bpf_spin_lock in 5.1
    - bounded loops are now supported in 5.3
  - https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md

How to run eBPF/XDP

- eBPF programming model



How to run eBPF/XDP



eBPF C Code

```
BPF_PERCPU_ARRAY(rxcnt, long, 1);
BPF_ARRAY(sequence, u64, 1);

int xdp_simple_responder(struct xdp_md *ctx) {
    void* data_end = (void*)(long)ctx->data_end;
    void* data = (void*)(long)ctx->data;
    struct ethhdr *eth = data;
    uint32_t key = 0;
    long *value;
    uint64_t nh_off;
    u16 h_proto;
    u32 ipproto;
    u64 zero = 0;
    nh_off = sizeof(*eth);

    if (data + nh_off  > data_end)
        return XDP_DROP;
        value = rxcnt.lookup(&key);
        if (value)
                *value += 1;

        uint64_t *counter = sequence.lookup(&key);
```

eBPF Python Loader

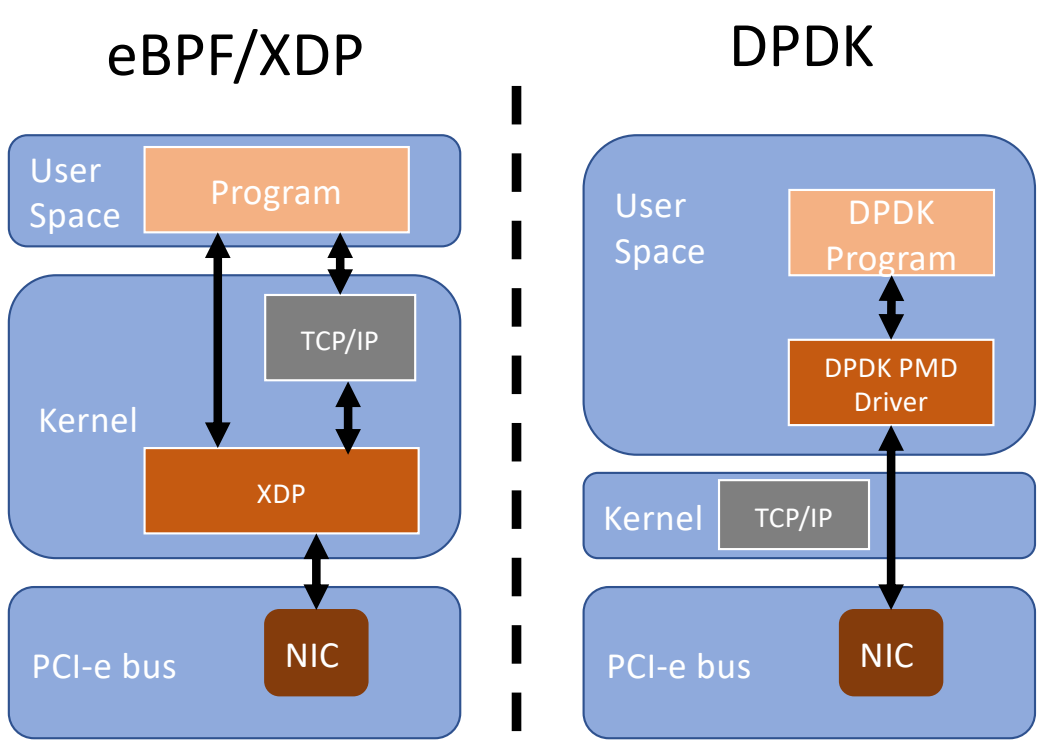
```
# load BPF program
b = BPF(src_file = "example1.c", cflags=["-w"], device=offload_device)


in_fn = b.load_func("xdp_simple_responder", mode, offload_device)

if mode == BPF.XDP:
        b.attach_xdp(device, in_fn, flags)
#else:

rxcnt = b.get_table("rxcnt")
prev = 0;
print("Printing redirected packets, hit CTRL+C to stop")
while 1:
    try:
        val = rxcnt.sum(0).value
        if val:
            delta = val - prev
            prev = val
            print("{} pkt/s".format(delta))
        print("ebpf running. hit CTRL+C to stop")
        time.sleep(1)
    except KeyboardInterrupt:
        print("Removing filter from device")
        break;

if mode == BPF.XDP:
    b.remove_xdp(device, flags)
#else:
```

Network Stack Bypassing

- Extended Berkeley Packet Filter
  - Push packet processing program into Kernel

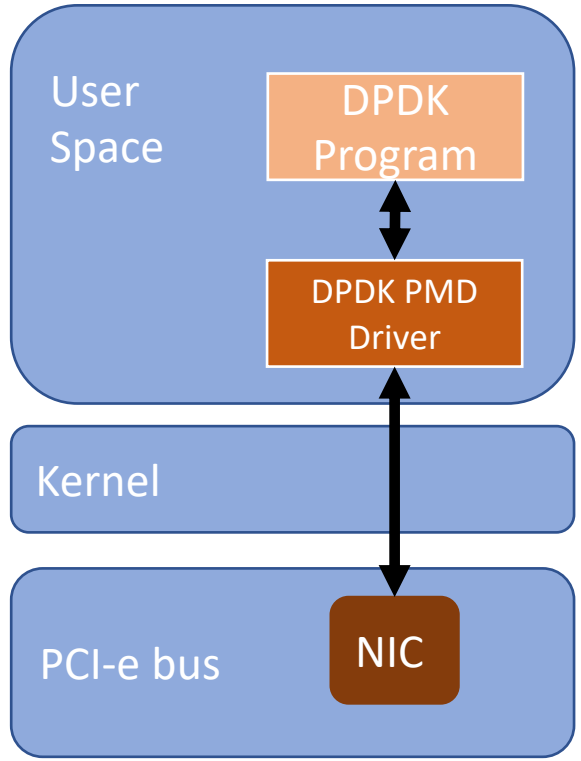
- Data Plane Development Kit
  - move packets directly into user space.

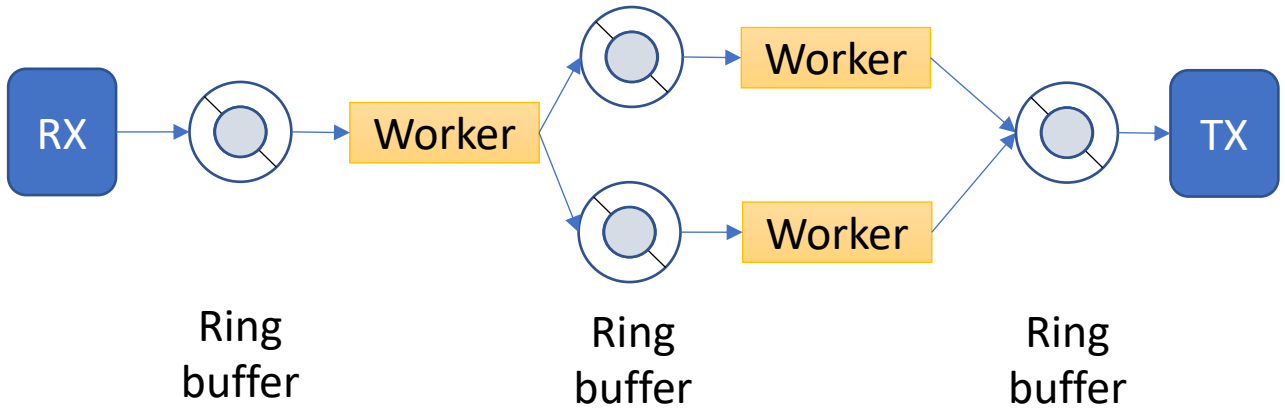
- Two philosophies, same goal



Data Plane Development Kit

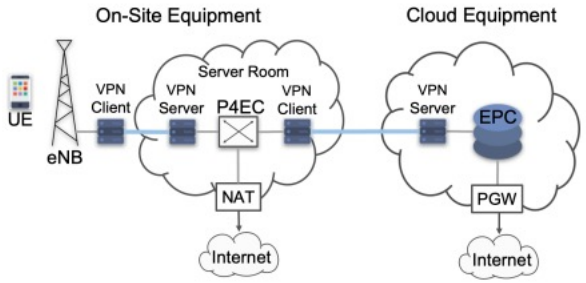
- Pros:
  - Kernel Bypassing
  - Poll Mode Driver
  - Multi-stage scalable architecture with ring buffers
- Cons:
  - Exclusive use of NIC port
  - CPU in polling mode.





Research Projects at Computer Systems Lab

- **P4EC**:Mobile Edge Computing



- Distributed consensus design with P4 switches and SmartNICs

- 2x Tofino P4 switches

- 5x SmartNICs