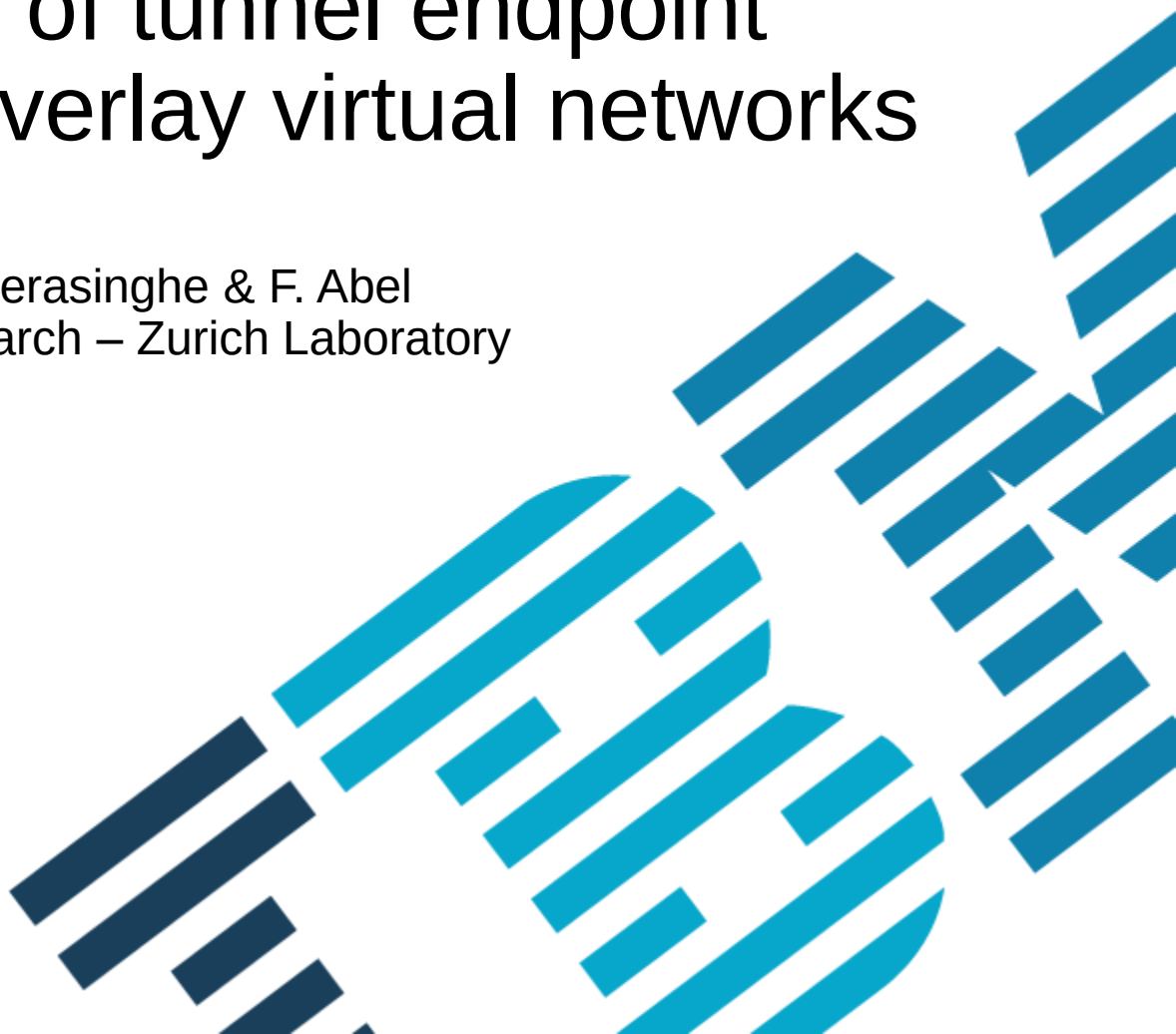
On the cost of tunnel endpoint processing in overlay virtual networks

J. Weerasinghe & F. Abel
IBM Research – Zurich Laboratory



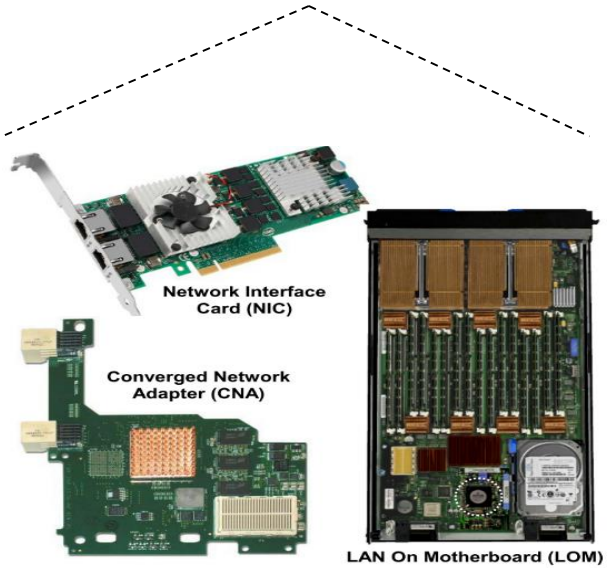
Outline

- Motivation
- Overlay Virtual Networks
  - background
  - tunnel endpoint
- Cost of Tunnel Endpoint Processing
- Proposal & Implementation of Acceleration
- Measurements
- Conclusions

Motivation

- Deployment of OVNs in Integrated NICs (iNICs)
  - area and power restricted

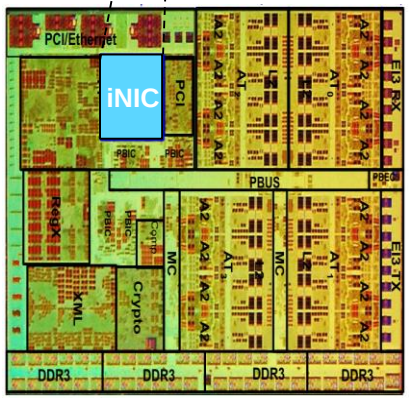
Rack- & Blade-Servers



Discrete NICs (PCIe- or CPU-attached)

Hyper-scale Servers

Micro Server



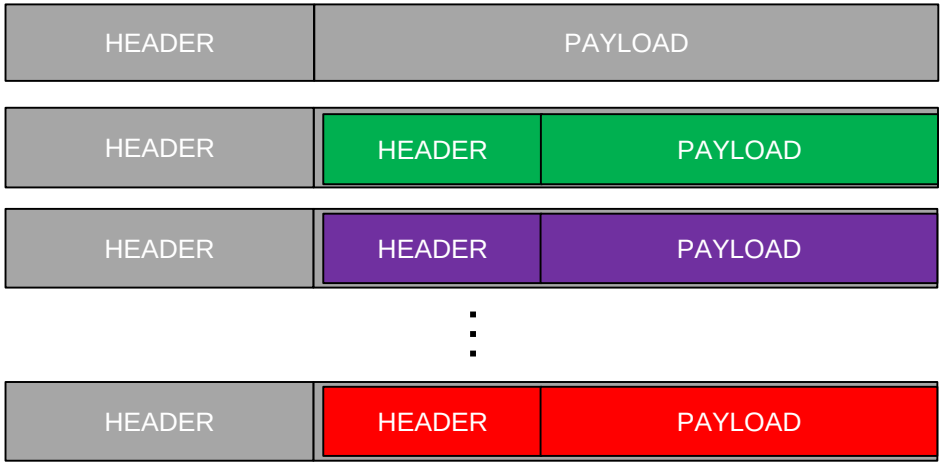
Integrated NIC (inside the CPU)

Overlay Virtual Networks (1/2)

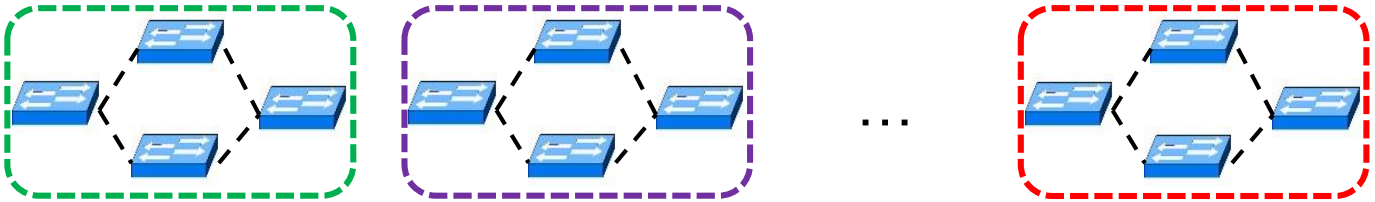
Single Physical Network

Packet Encapsulation

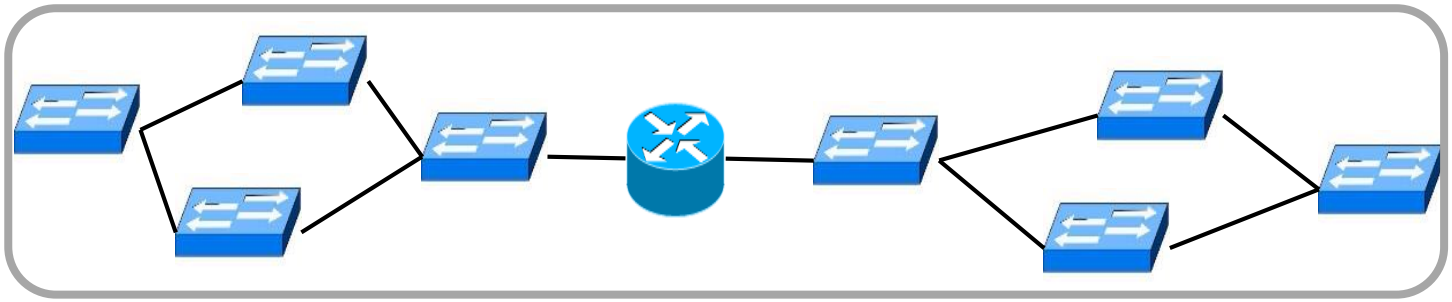
Millions of Virtual Networks



Millions of Virtual L2 Networks/ Overlay Networks

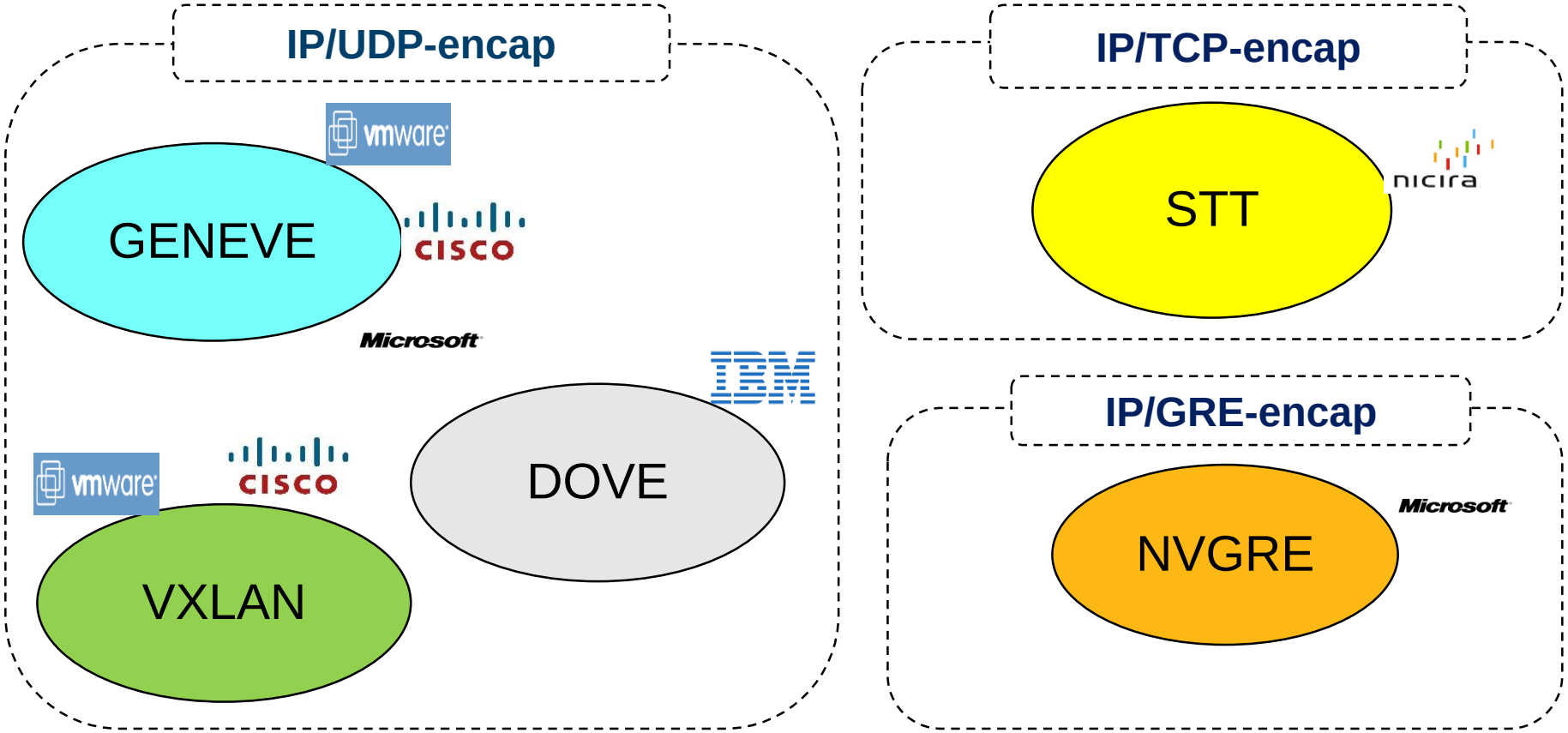


Single Physical Network/ Underlay Network (**Both L2 & L3**)



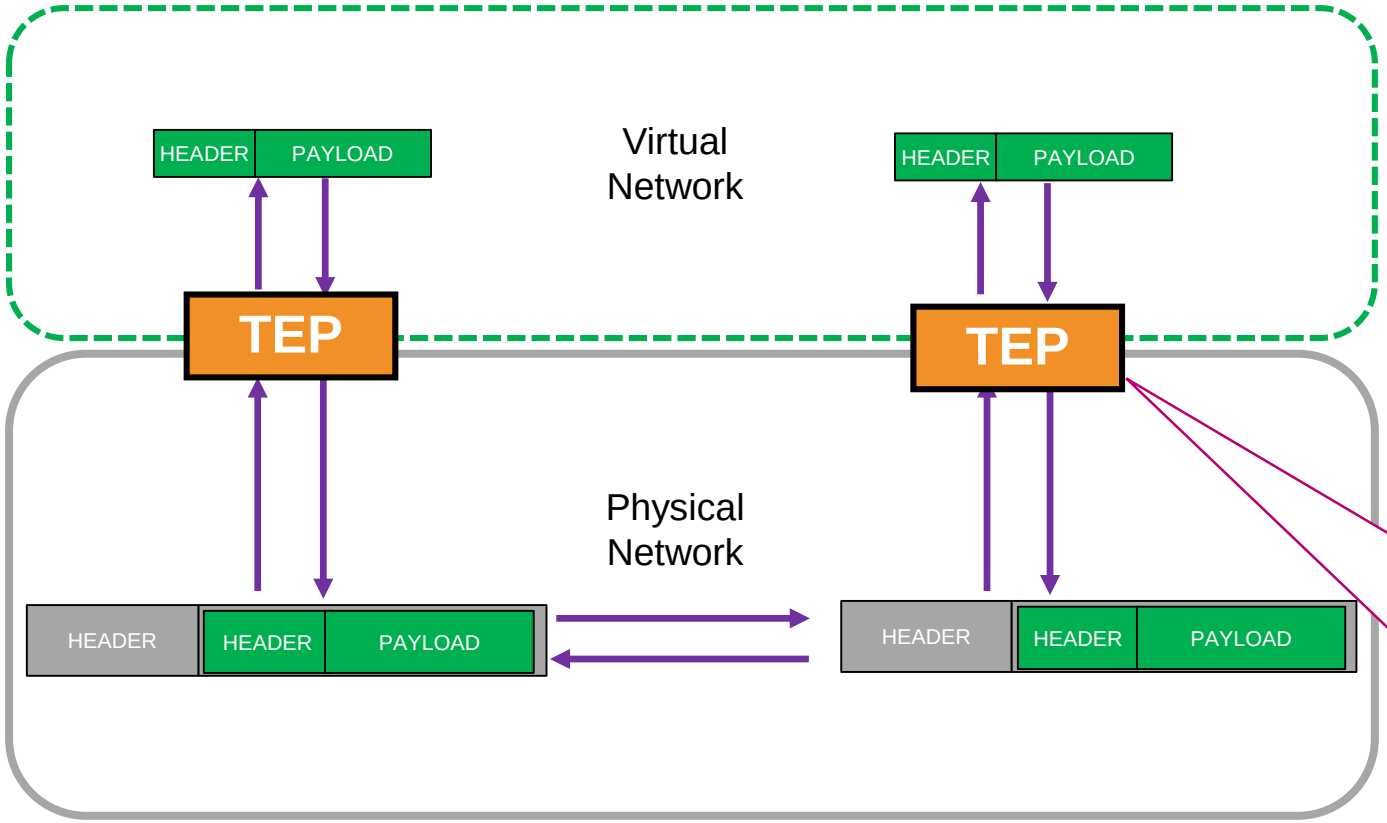
Overlay Virtual Networks (2/2)

- Many Flavors
  - different encapsulation protocols



Tunnel Endpoint (TEP)

- Place where packets are encapsulated and de-capsulated
- Usually an IP interface (L4 depends on encap protocol)

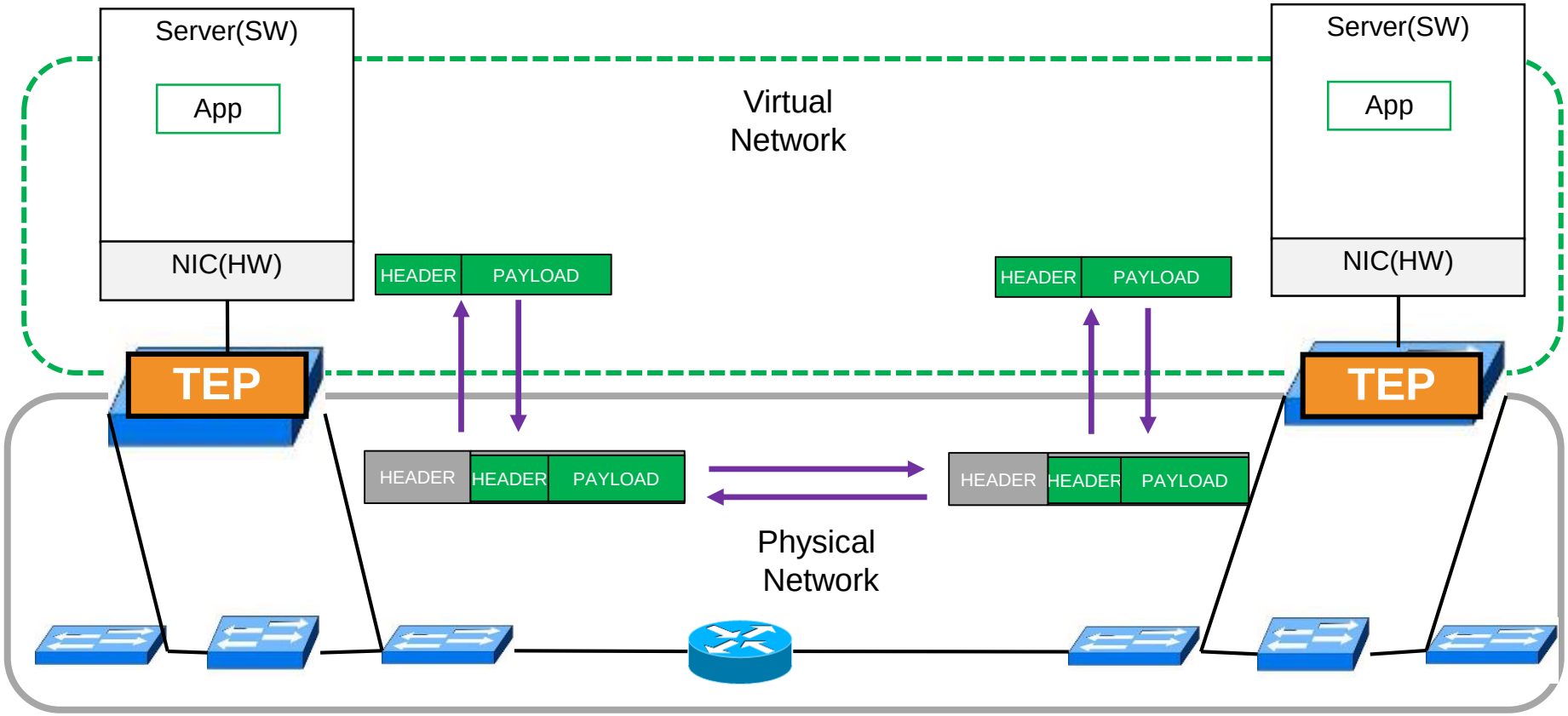


Can be implemented in:

(a) HW Switch
(b) NIC
(c) SW

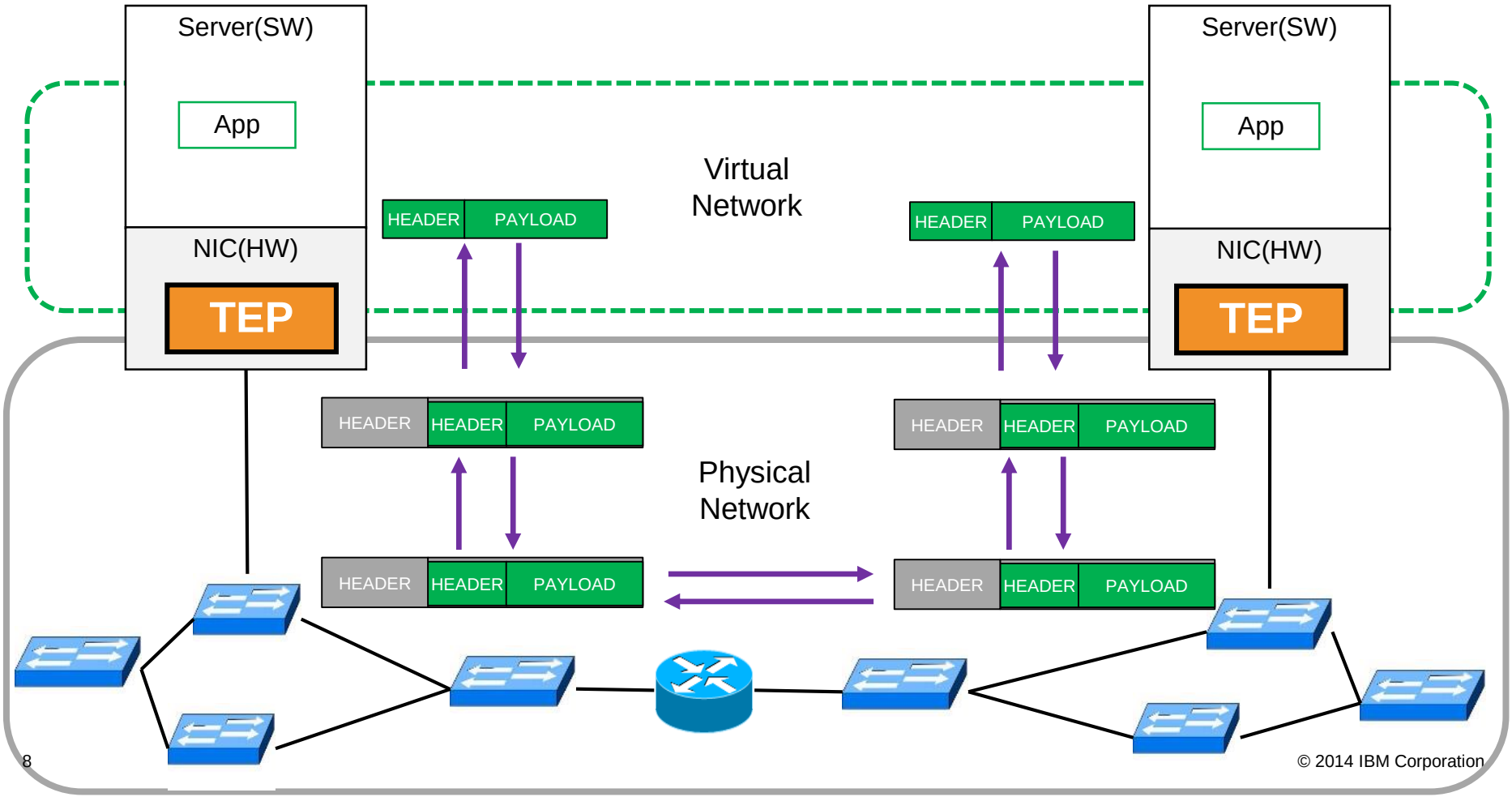
a) TEP in HW Switch: Pros & Cons

- Performance is good
- But not scalable
  - isolation of traffic between App ⇔ TEP
  - MAC table explosion



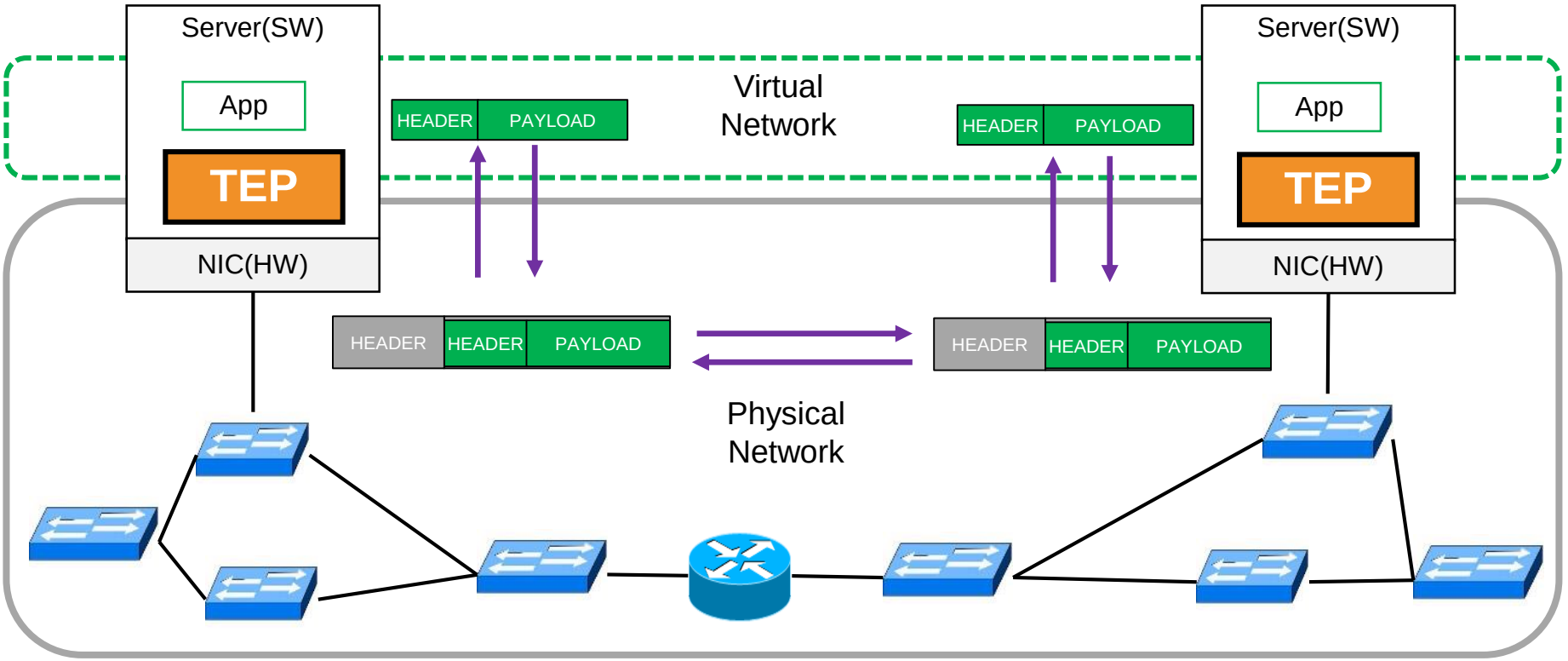
b) TEP in NIC: Pros & Cons

- Performance is good
- But not scalable
  - isolation of traffic between App ⇔ TEP
  - MAC table explosion



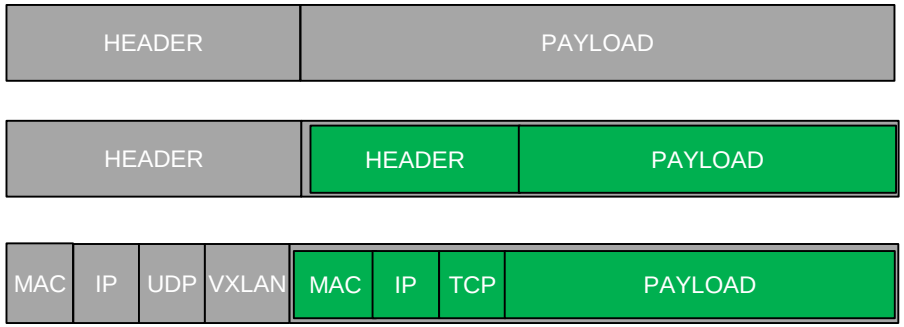
c) TEP in SW: Pros & Cons

- Scales well
- But longer code-path in SW degrades performance



Approach On Improving SW TEP Performance

- OVN
  - VXLAN
- Application Protocol
  - TCP/IP
    - a widely used application protocol
- Rx Path
  - critical part of packet processing
    - due to lack of prior knowledge
- SW TEP
  - analyze Linux implementation
  - assess the Cost
- Identify Functions to be Accelerated
- Accelerate the Identified Functions

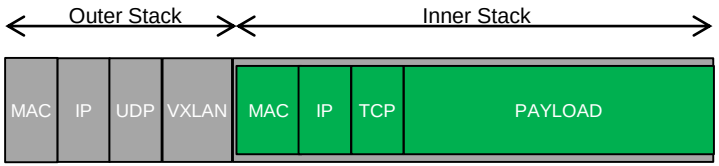


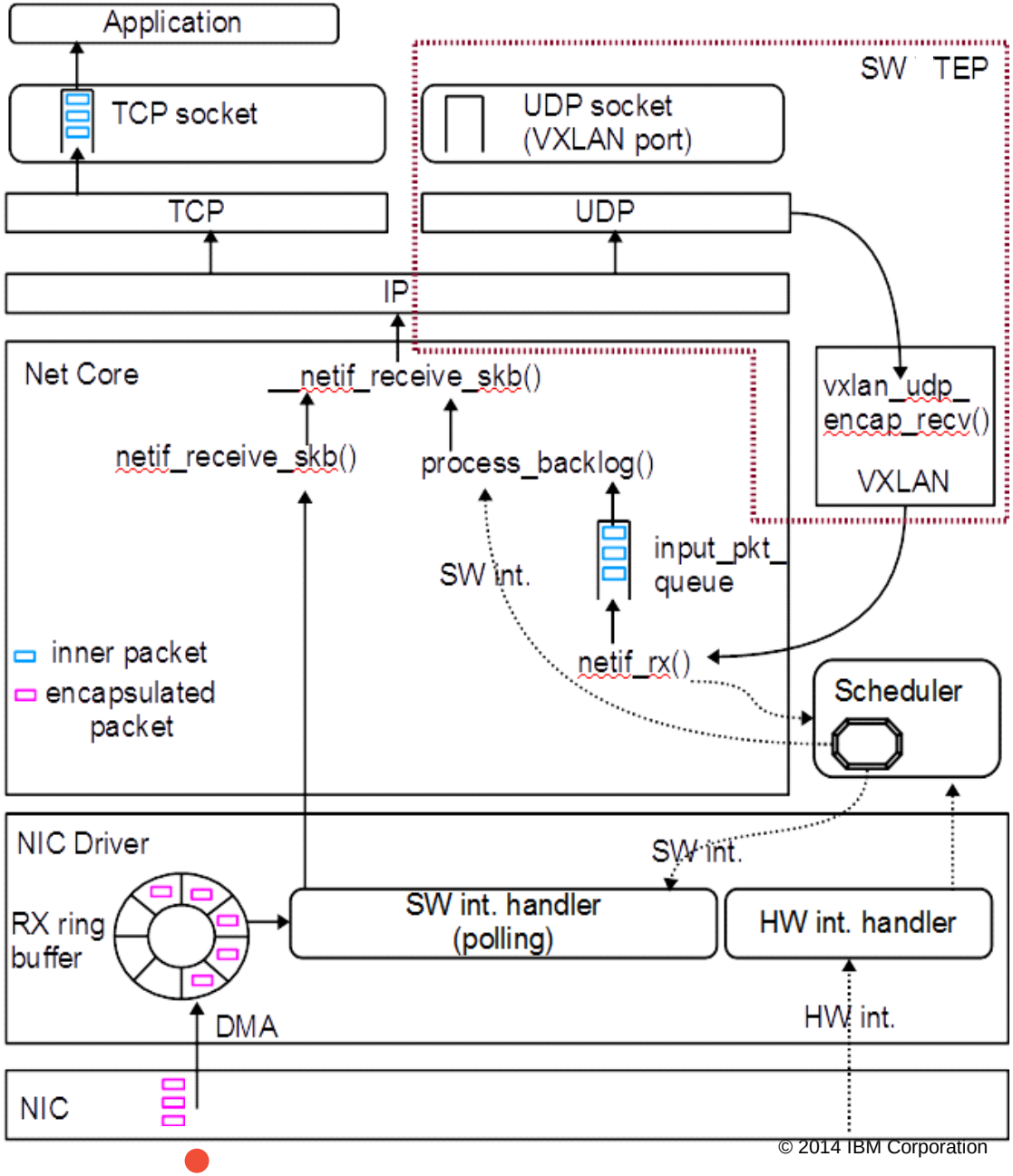
VXLAN-encapsulated TCP/IP packet

VXLAN-encapsulated TCP/IP Packet

- Path in the Linux Network Stack

each packet has to travel twice in the stack





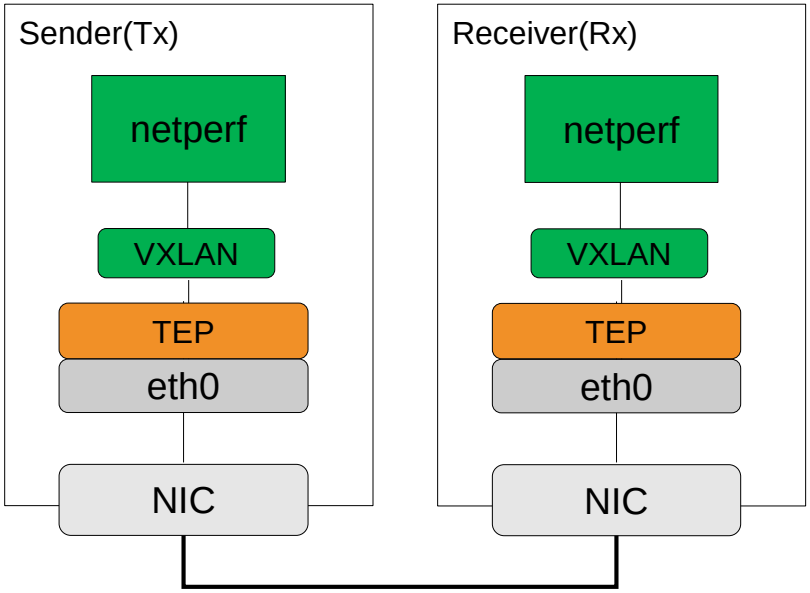
Assessing the Cost of TEP Processing (1/4)

- Experimental Setup
  - Linux on bare metal
  - sender and receiver
    - netperf-based
    - connected back-to-back
- Measurements
  - clock cycles (for TEP processing)
  - BW
  - latency

Specification	Sender	Receiver
CPU	Intel Core i7 3820	Intel Core i7-950
Cores/CPU Threads	4/8	1/1
Clock Speed[GHz]	3.6	3.07
Memory[GB]	32	12
Cache L1/L2[KB]/L3[MB]	32/256/10	32/256/8
Network Interface Card	10G NIC (a.)	10G NIC (a.)
Linux Kernel/Distribution	3.12.18/Fedora 20	3.12.18/Fedora 20
TCP Socket Size[KB]	64	84

a. A FPGA-based (Altera Stratix IV GX) NIC implemented on a PLDA XpressGX4-LP PCIe card

Specification of the Experimental Setup



Experimental Setup

Assessing the Cost of TEP Processing (2/4)

- Clock Cycle Measurement
  - fine grained instrumentation of the code using the time stamp counter
- Procedure

Linux Kernel Network Source Code

M1 = Measure()
M2 = Measure()

M3 = Measure()

Code Segment of Interest

M4 = Measure()

Measurement overhead = M2–M1

Clock cycles spent on executing the code w/ measurement overhead = M4-M3

Clock cycles spent on executing the code = (M4-M3)-(M2-M1)

Assessing the Cost of TEP Processing (4/4)

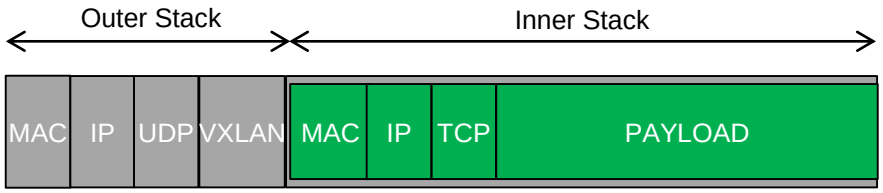
- CPU Clock Cycles for VXLAN Packet Processing

Stack	Layer/ Function	Sub Function	Clock Cycles	Total	% of MTU-Size Packet
Outer	Net Core		120	1224	21%
	L3 (IP)		668		
	L4 (UDP)		180		
	VXLAN		256		
Inner	Net Core		92	1604	27%
	L3 (IP)	Checksum	24		
		Other	228		
	L4 (TCP)	Checksum	608		
		Other	652		

21% overhead

10% overhead

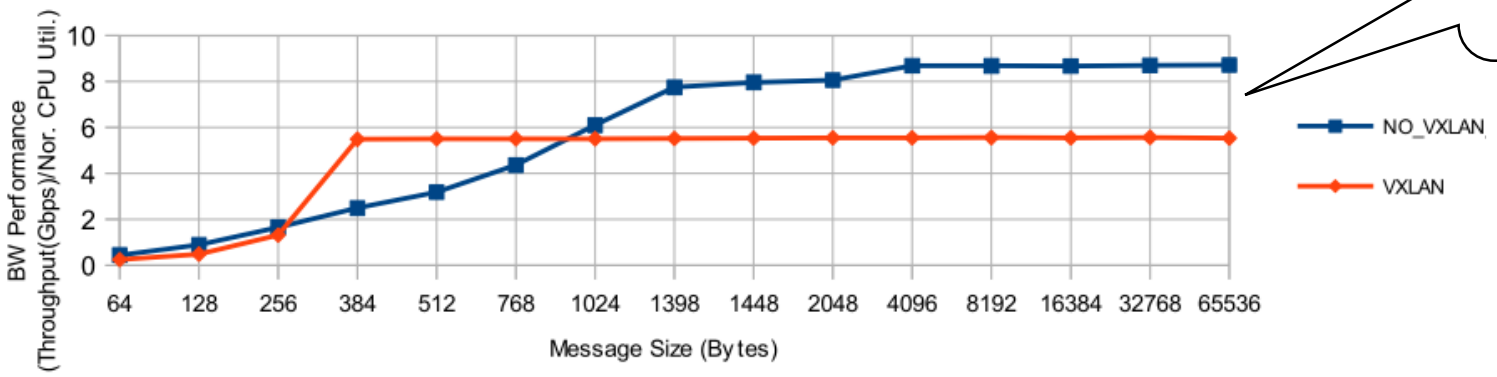
Number of Clock Cycles Spent on Outer- and Inner-stack Processing



Format of VXLAN-encapsulated TCP/IP packet

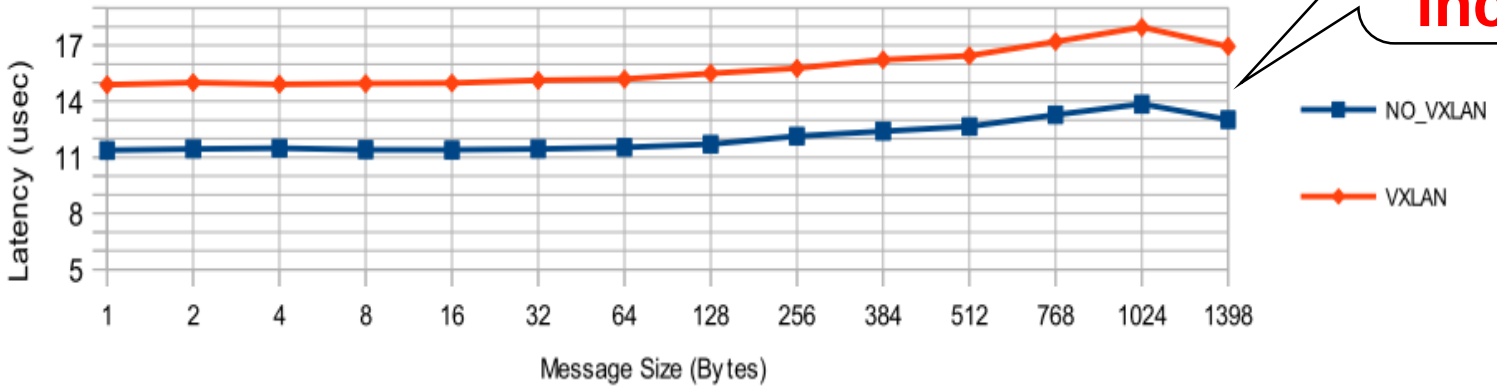
Assessing the Cost of TEP Processing (3/4)

- Bandwidth
  - Netperf TCP_STREAM
  - BW performance = data rate (Gbps)/ CPU utilization



31.8% BW drop

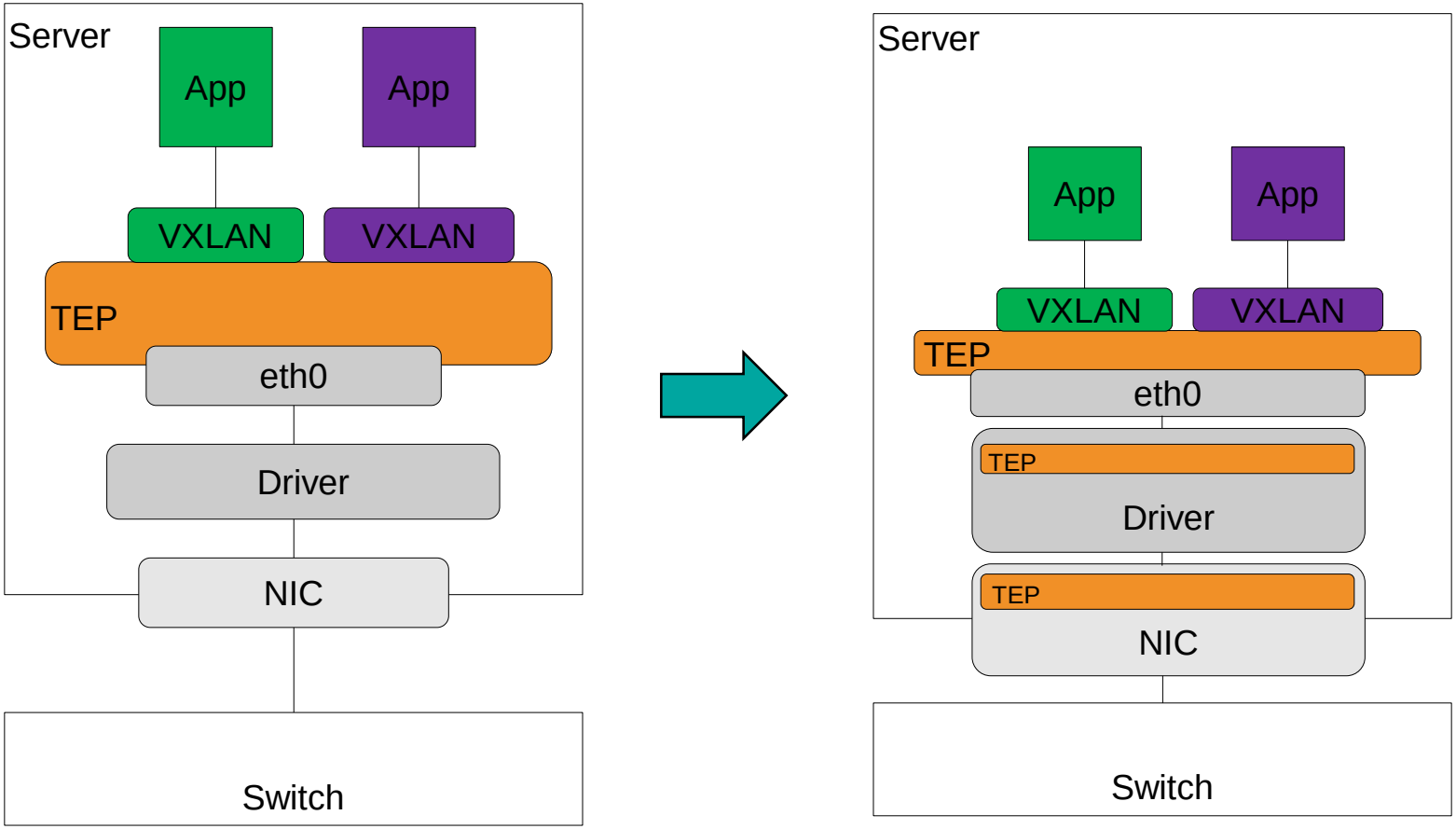
- Latency
  - Netperf TCP_RR
  - RTT/2



32.5% latency increment

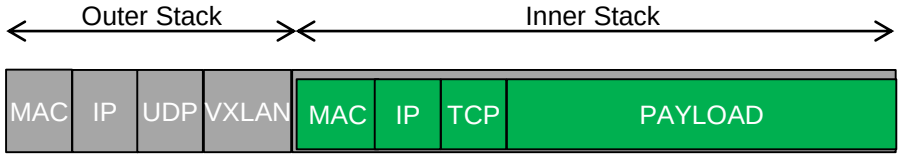
Proposed Acceleration (1/3)

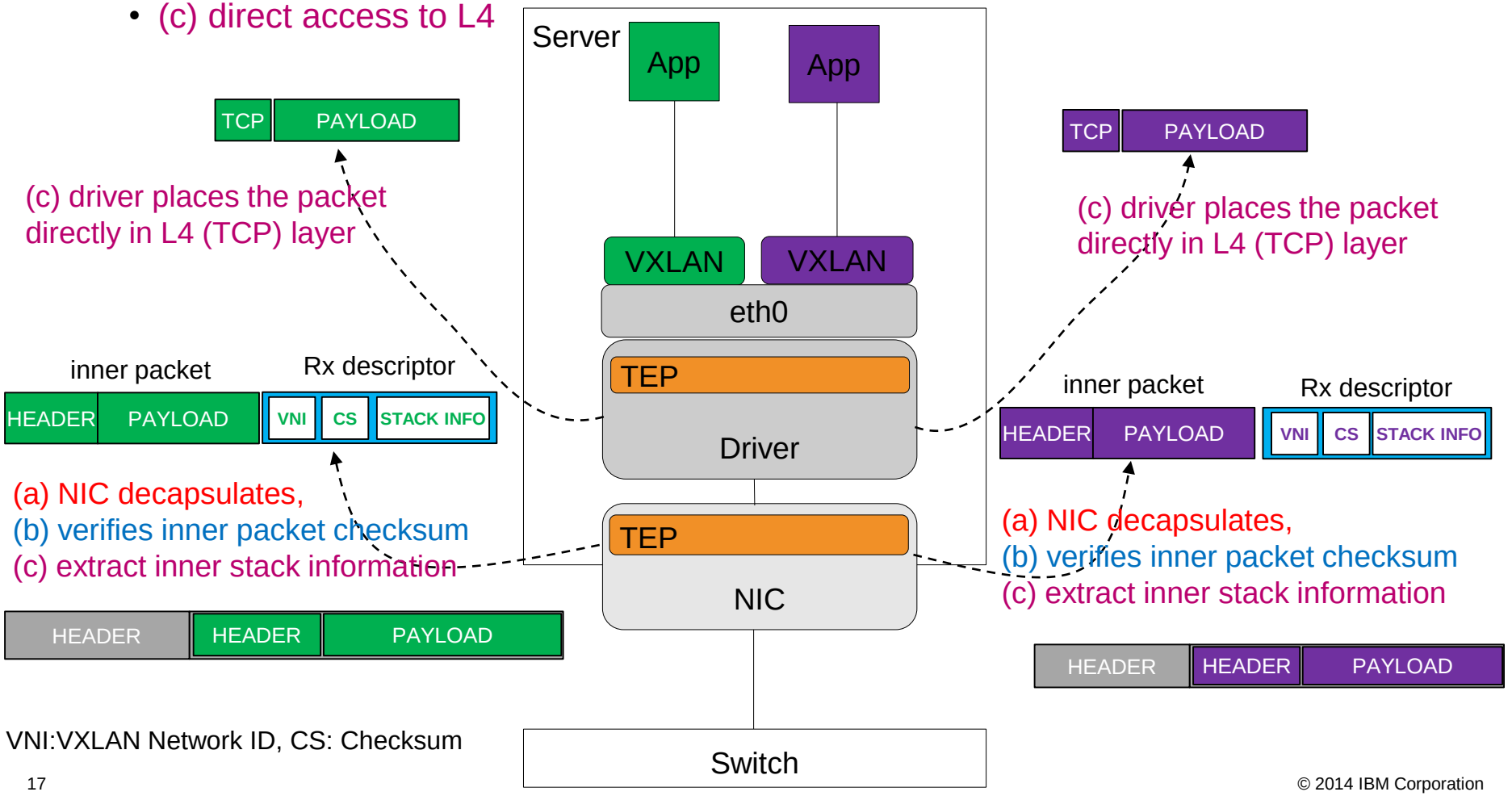
- SW TEP ➔ Hybrid (SW & HW) TEP
  - part of Rx path SW TEP processing moved to NIC HW & Driver
  - Tx path TEP processing not changed



Proposed Acceleration (2/3)

- Stack Acceleration
  - outer stack acceleration (OSA)
    - (a) packet de-capsulation in NIC
  - inner stack acceleration (ISA)
    - (b) checksum in NIC
    - (c) direct access to L4





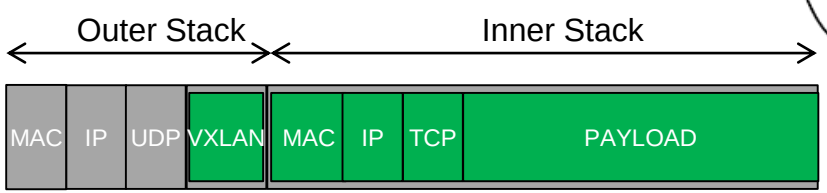
VNI:VXLAN Network ID, CS: Checksum

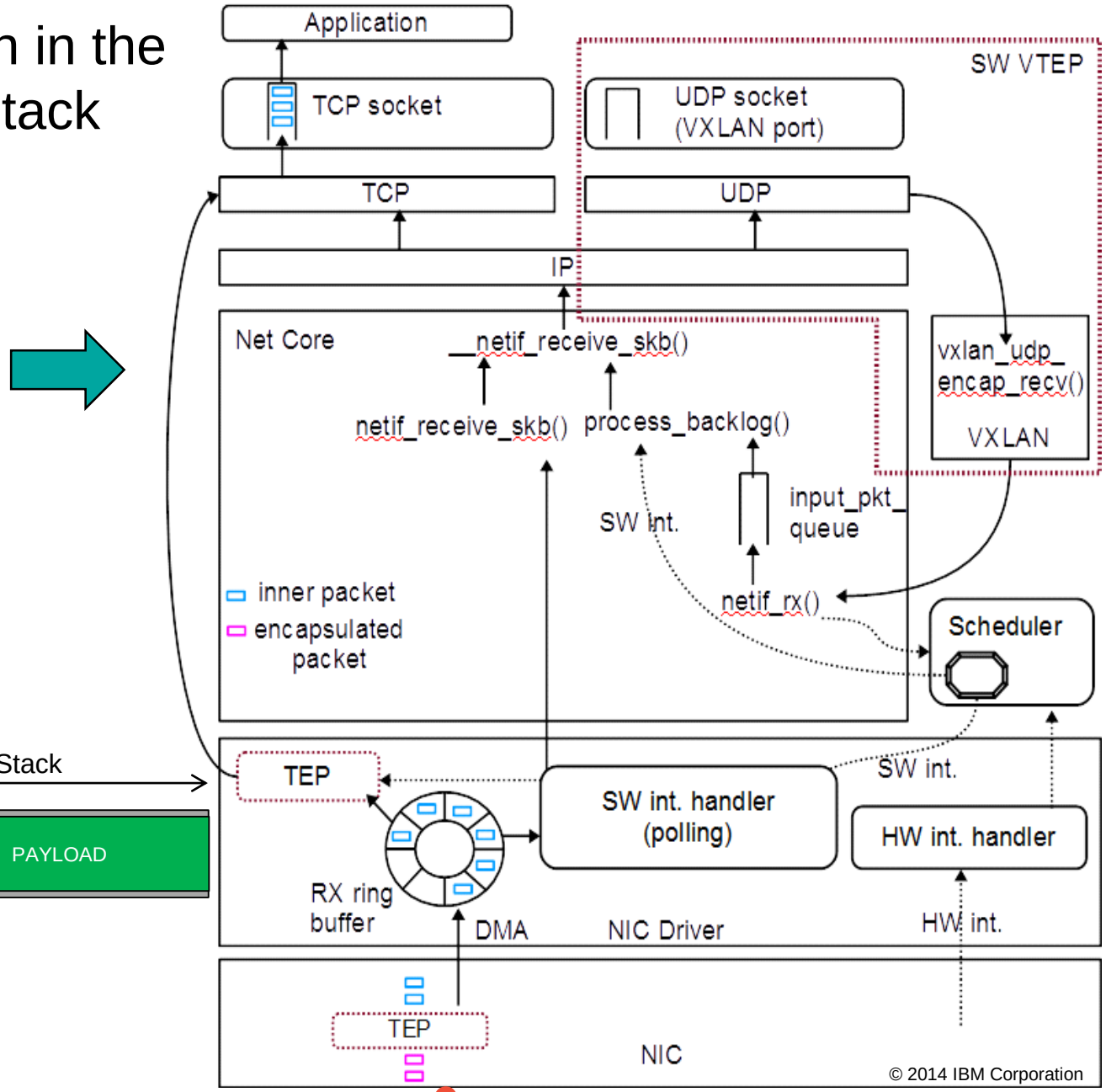
Proposed Acceleration (3/3)

- Accelerated Path in the Linux Network Stack

(a) each packet travels only once in the stack

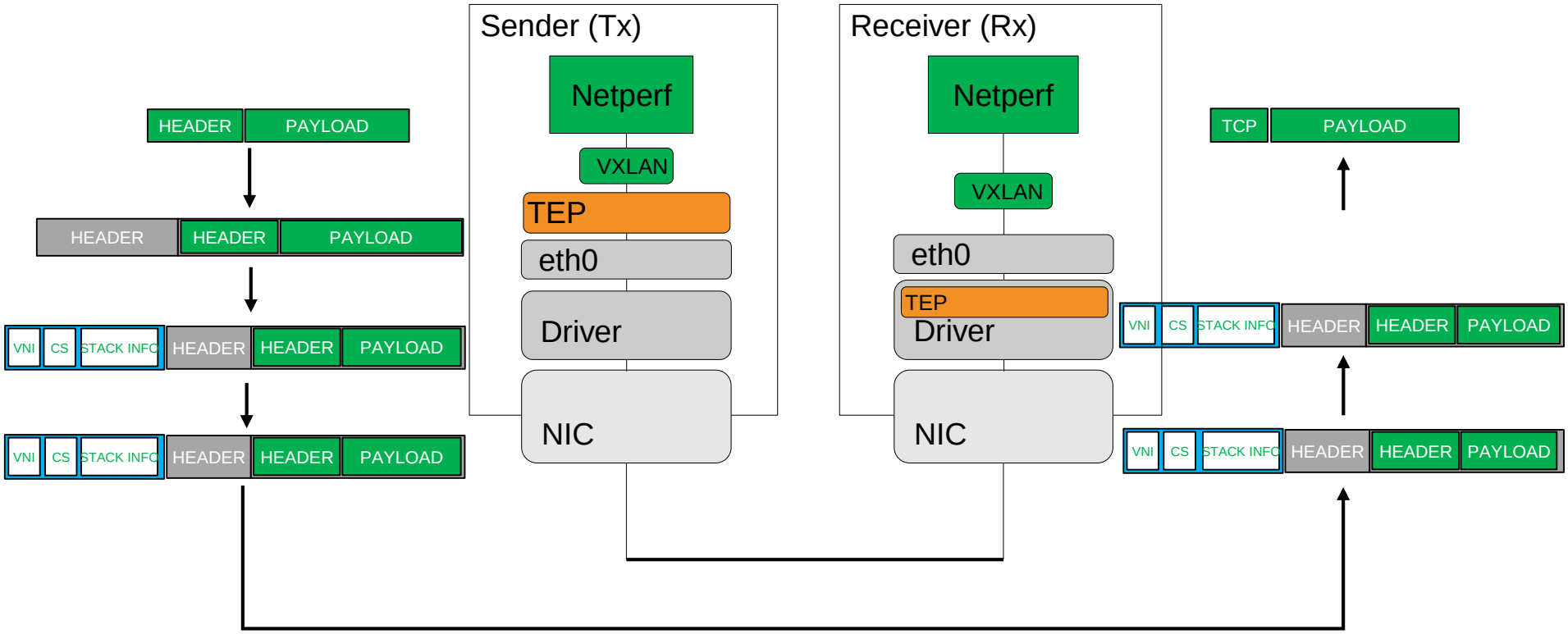
(b) L2 and L3 are bypassed





Implementation

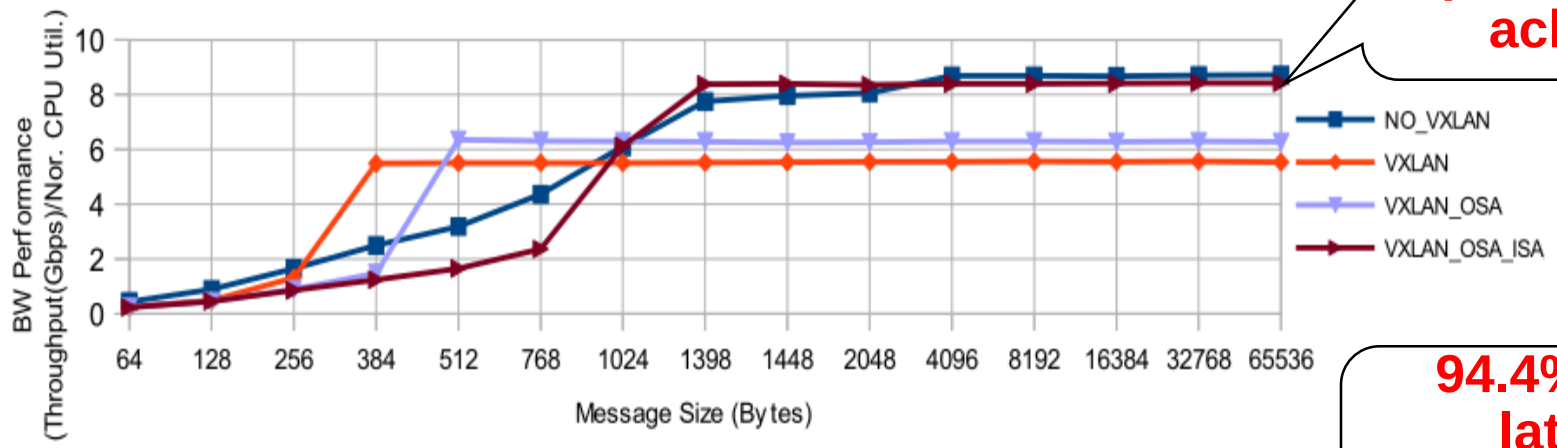
- @Tx driver
  - meta-data is generated (in real implementation this happens in Rx NIC)
  - and prepended to the packet
- @Rx driver
  - meta-data is removed
  - and packet is decapsulated (in real implementation this happens in Rx NIC)



VNI:VXLAN Network ID, CS: Checksum

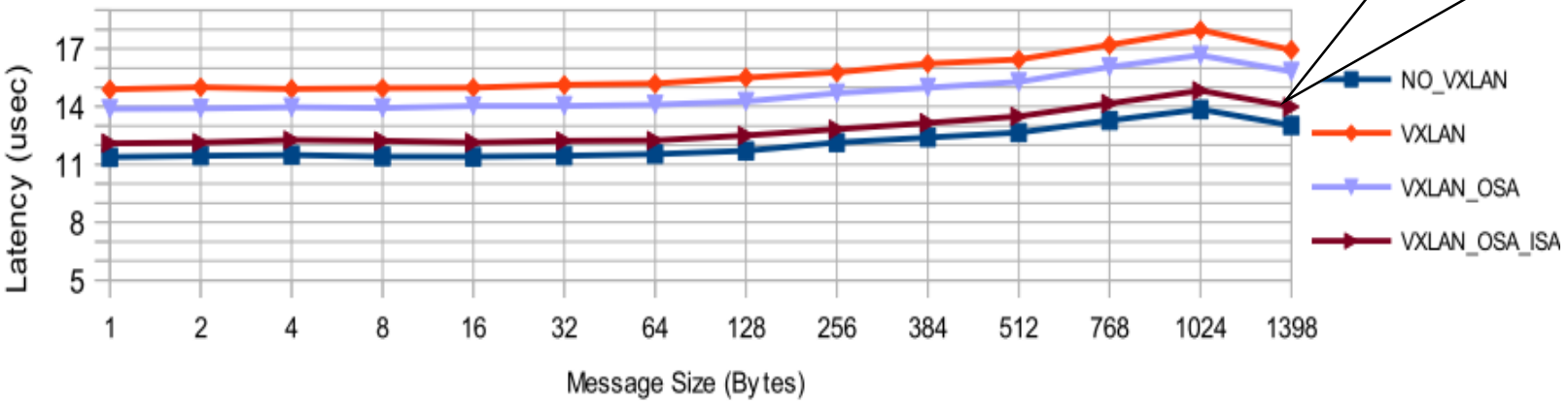
Results

- Bandwidth



97.1% of the BW performance achieved

- Latency

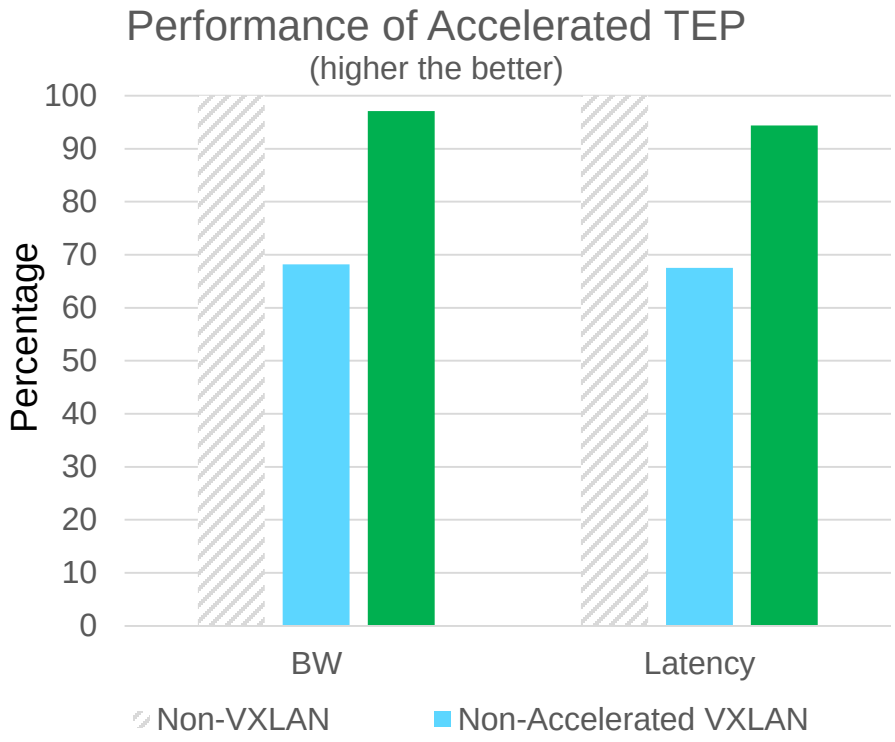


94.4% of the latency performance achieved

OSA: Outer Stack Acceleration
ISA: Inner Stack Acceleration

Conclusion

- SW Tunnel Endpoint
  - supports all OVN requirements
  - but performance is degraded
  - cost
    - VXLAN adds **21%** of CPU cycles to the processing of a MTU-size packet
    - BW performance is dropped by **31.8%**
    - latency is increased by **32.5%**
- Accelerated Tunnel Endpoint
  - light-weight stack acceleration
  - achieved performance
    - **97.1%** of BW
    - **94.4%** of latency
- Future Work
  - add OVN support to integrated NICs



THANKS