Revisit the IP stack in Linux with Network Virtualization

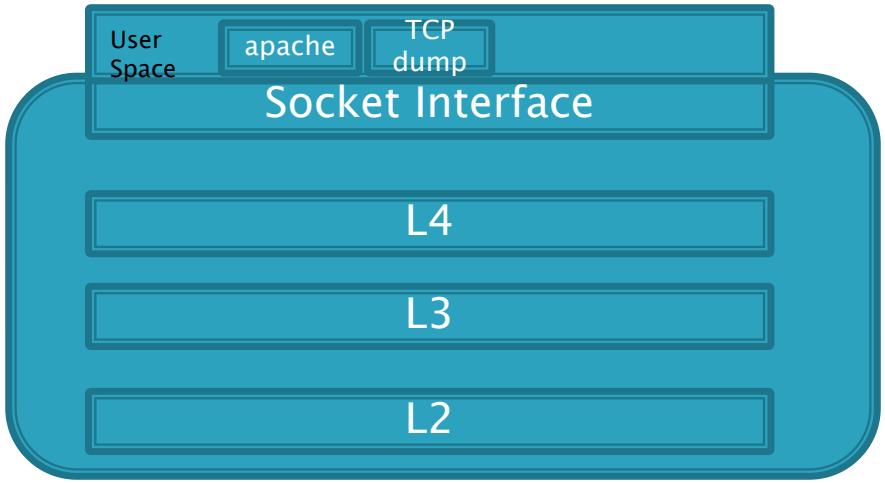
Jun (Jim) Xu
jun.xu@huawei.com
Principal Engineer,
Futurewei Technologies, Inc.

A Quick Survey

- Linux
- KVM/QEMU
- Switch/Router
- NFV

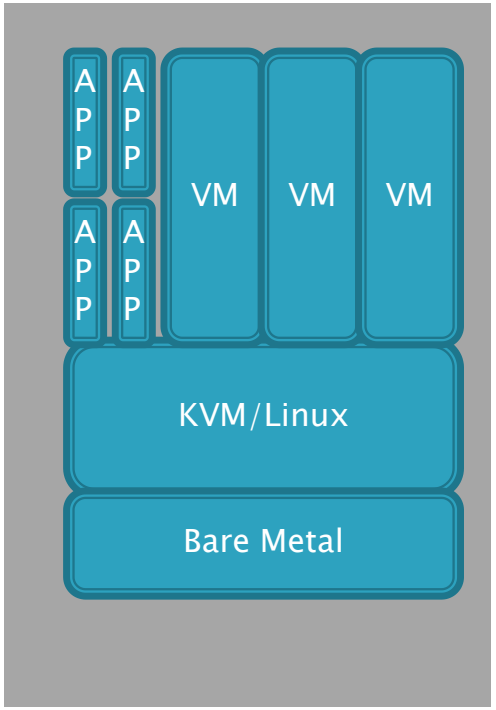
Linux Before Hypervisor

- Linux IP stack in Kernel
- All Applications will communicate via socket
- Limited raw socket applications
- A Perfect world (really ?)



Virtualization in Linux

- ‘**KVM (Kernel-based Virtual Machine)** is a virtualization infrastructure for the Linux kernel that turns it into a hypervisor, which was merged into the Linux kernel mainline in February 2007” *
  - Supports multiple architectures
  - Common use in network areas and ISP/SP
- KVM inherits majority of Linux Kernel functions, including its IP stack

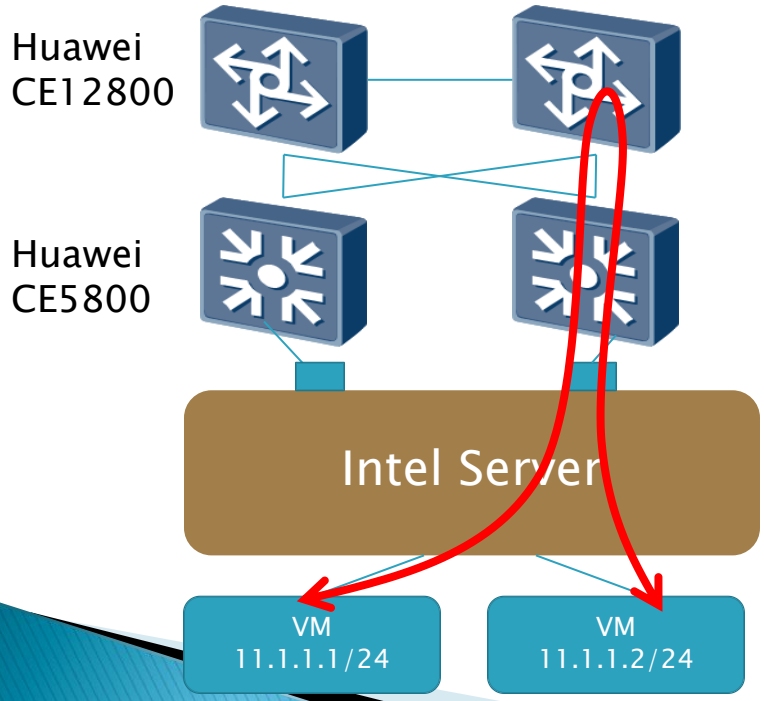


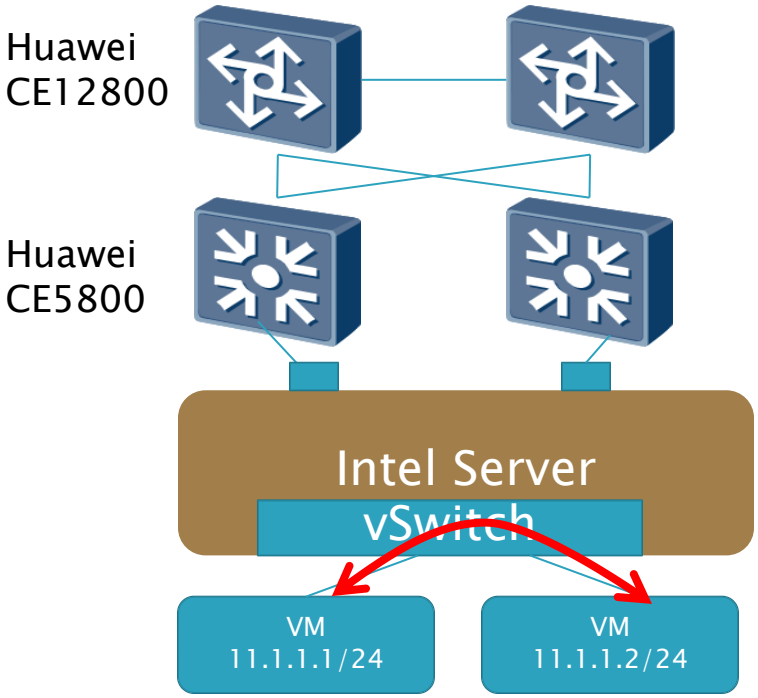
* From wikipedia

Big Picture – Network Virtualization in DC

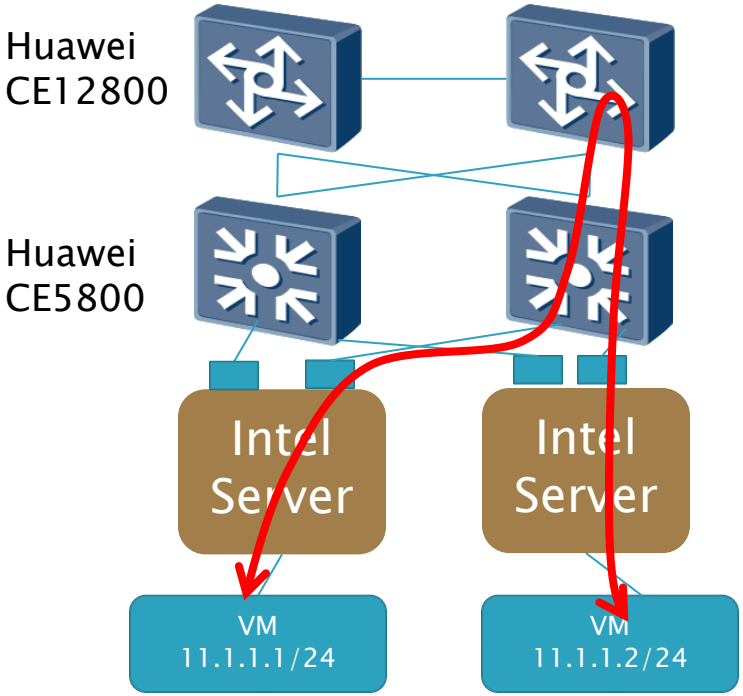
- Traffic pattern in DC:
  - Traffic across VM within the host
  - Traffic across hosts
  - Traffic aggregate to core and goes to Edge

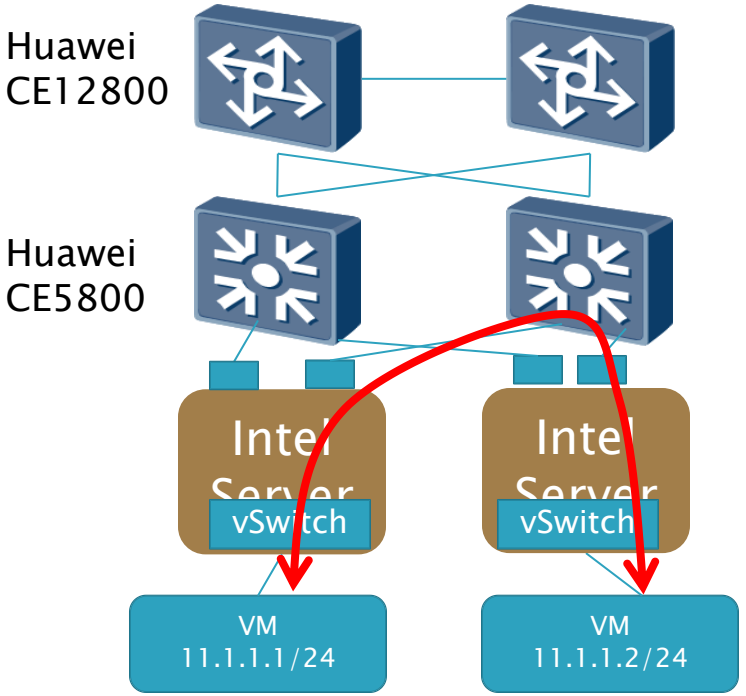
- Most traffics are the first two types (east-west)
- To handle the first two cases, virtual switch is introduced





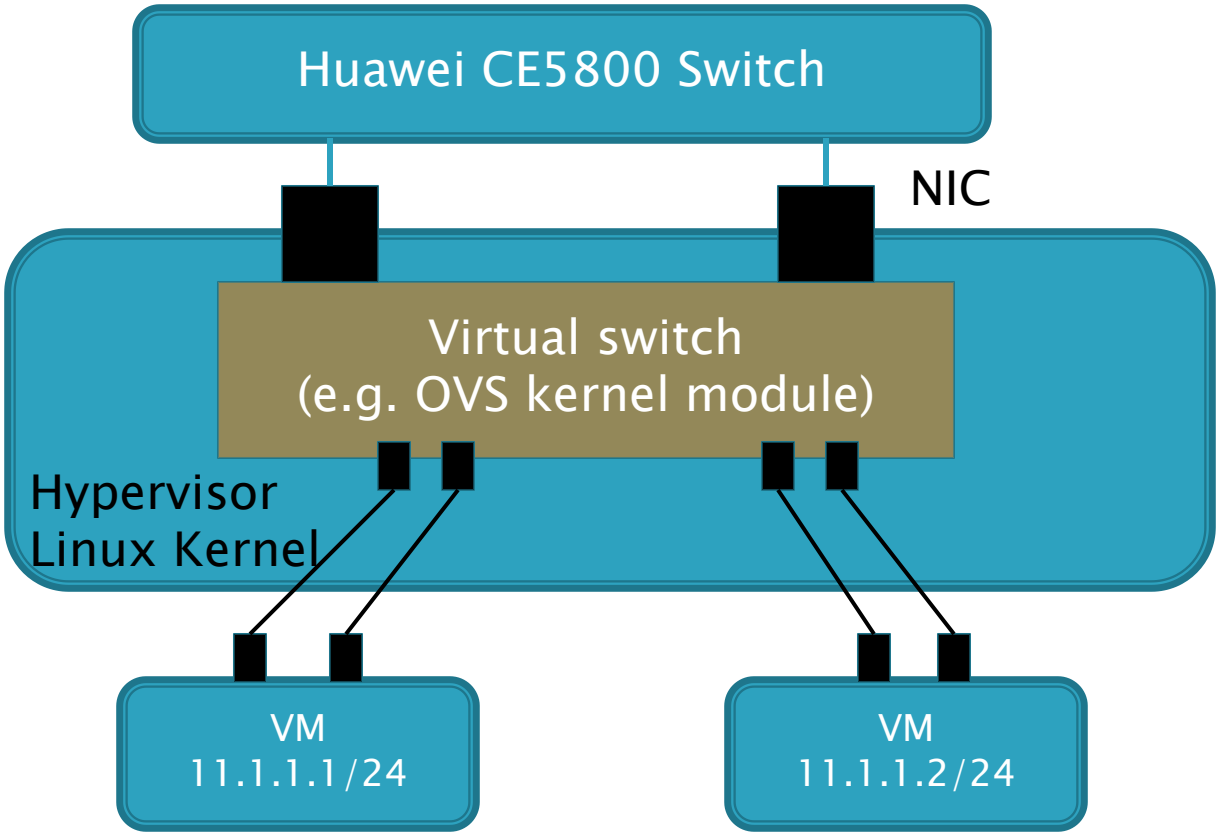
Big Picture – Network Virtualization in DC (cont.)





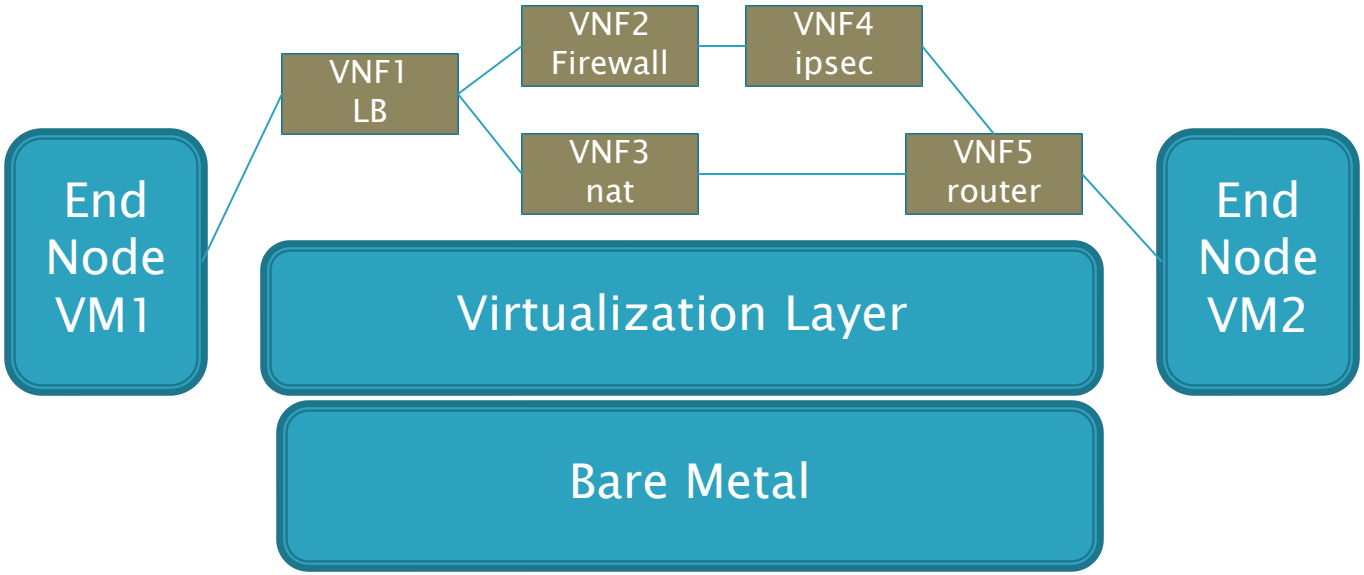
- VxLAN, STT, NVGRE can be used in virtual switch
- Distributed Router can be deployed in the host as well

Virtual Switch Details



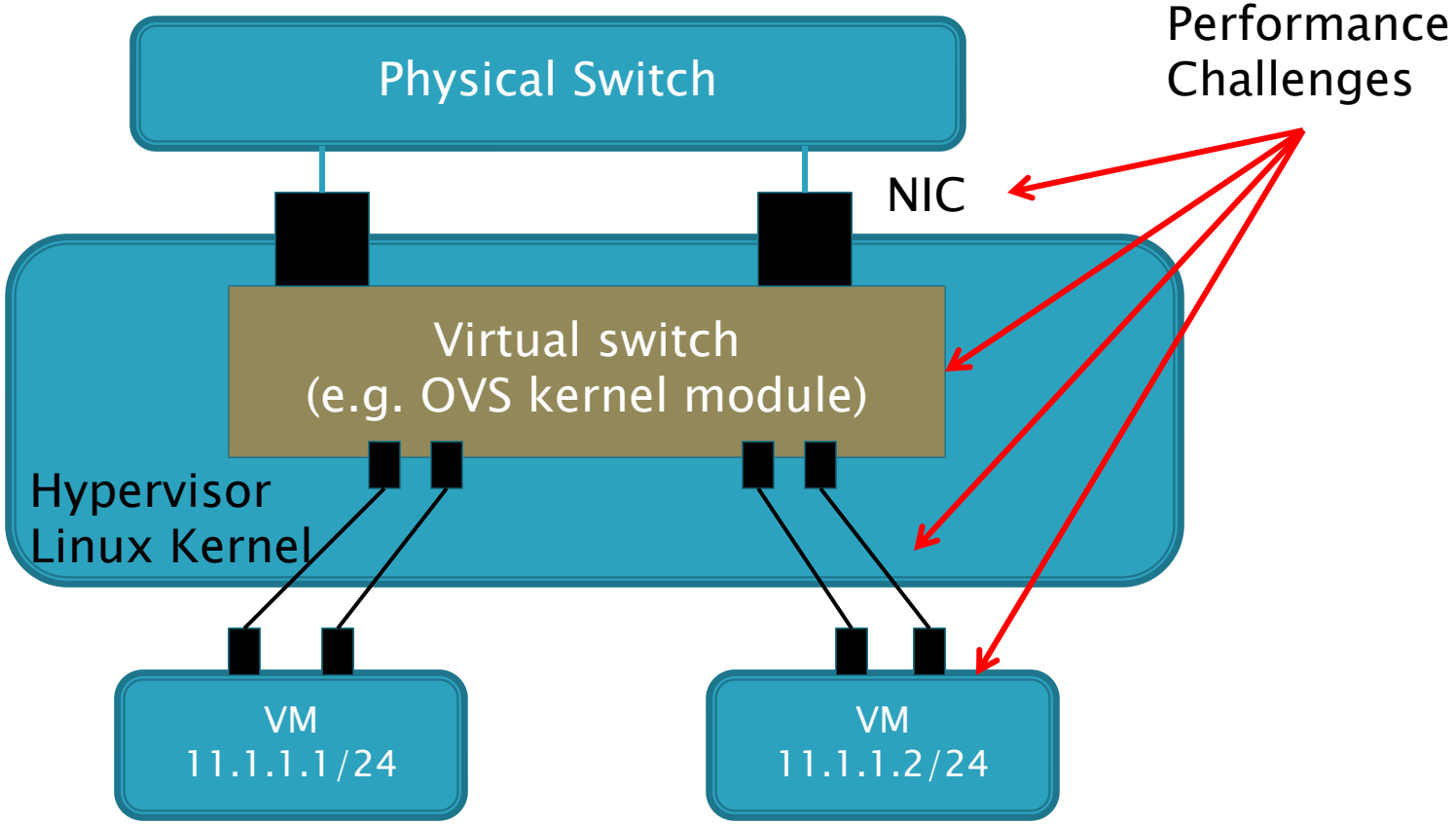
- There is one open source virtual switch – OVS
- Current OVS suitable for endpoint virtualization
- A Perfect World! (really ?)

Network Functions Virtualization (NFV)



- VNFs will be executed in VMs (for now).
- The packet performance along with the functions of the virtual switches will put significant impact on the success of the NFV
- These introduce new challenges…
- Reference to www.etsi.org about NFV

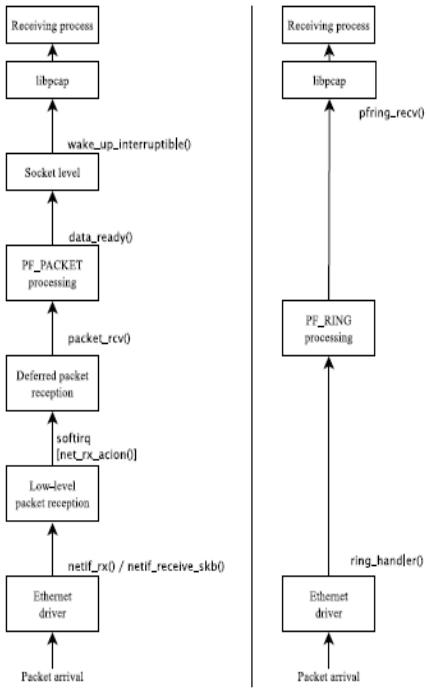
Virtual Switch Challenges



Issue One: Heavy Host IP Stack

File	Function/description	time ns	delta ns
user program	`sendto` system call	8	96
uipc_syscalls.c	`sys_sendto`	104	
uipc_syscalls.c	`sendit`	111	
uipc_syscalls.c	`kern_sendit`	118	
uipc_socket.c	`sosend`	—	
uipc_socket.c	`sosend_dgram` sockbuf locking, mbuf allocation, copyin	146	137
udp_usrreq.c	`udp_send`	273	
udp_usrreq.c	`udp_output`	273	57
ip_output.c	`ip_output` route lookup, ip header setup	330	198
if_ethersubr.c	`ether_output` MAC header lookup and copy, loopback	528	162
if_ethersubr.c	`ether_output_frame`	690	
ixgbe.c	`ixgbe_mq_start`	698	
ixgbe.c	`ixgbe_mq_start_locked`	720	
ixgbe.c	`ixgbe_xmit` mbuf mangling, device programming	730	220
–	on wire	950	

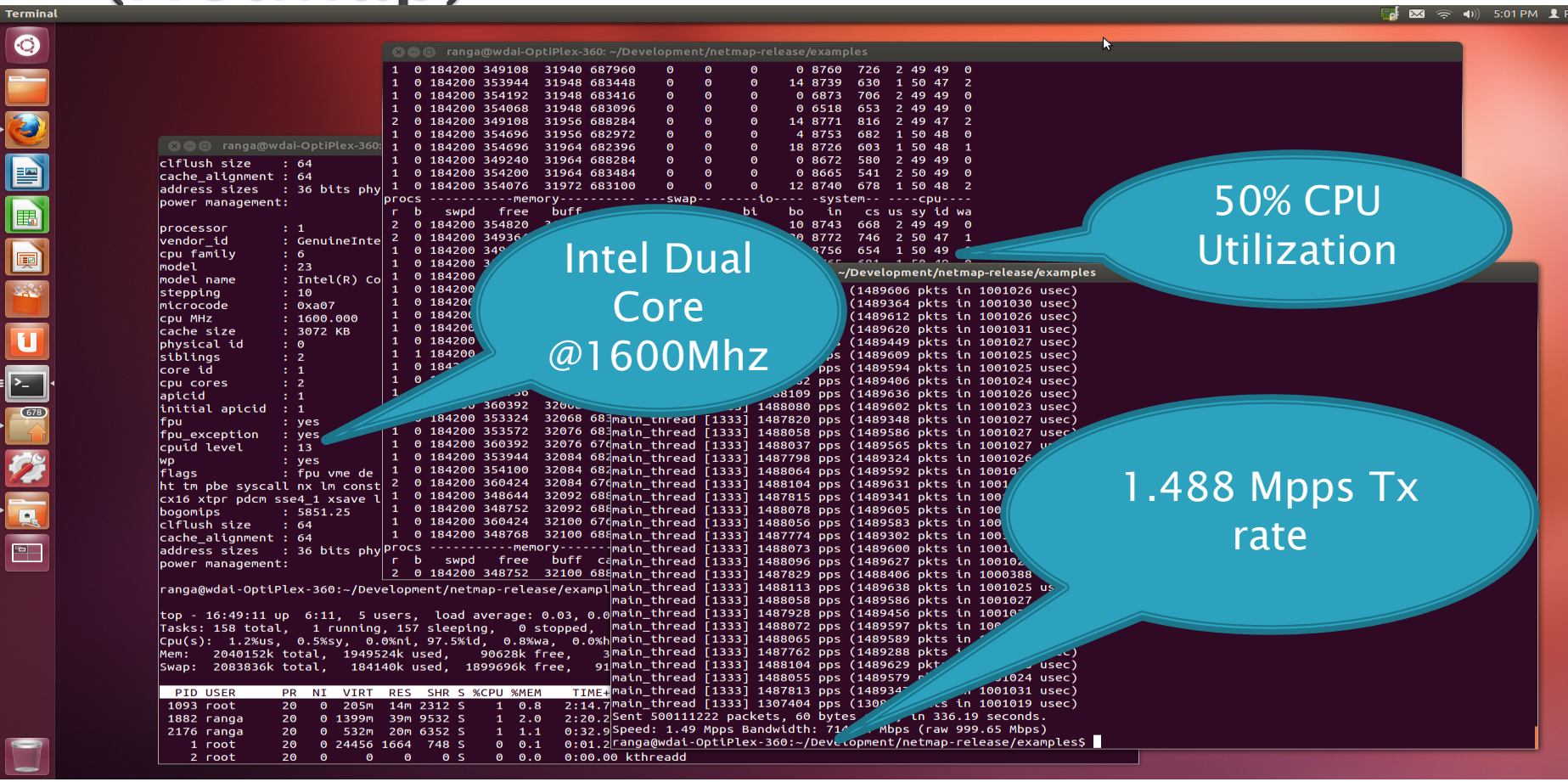
Figure 2: The path and execution times for `sendto()` on a recent FreeBSD HEAD 64-bit, i7-870 at 2.93 GHz + Turboboost, Intel 10 Gbit NIC and ixgbe driver. Mea-



From: Lothar Braun, Alexander Didebulidez, etc., *"Comparing and Improving Current Packet Capturing Solutions based on Commodity Hardware"* in Internet Measurement Conference, 2010

http://conferences.sigcomm.org/imc/2010/papers/p206.pdf

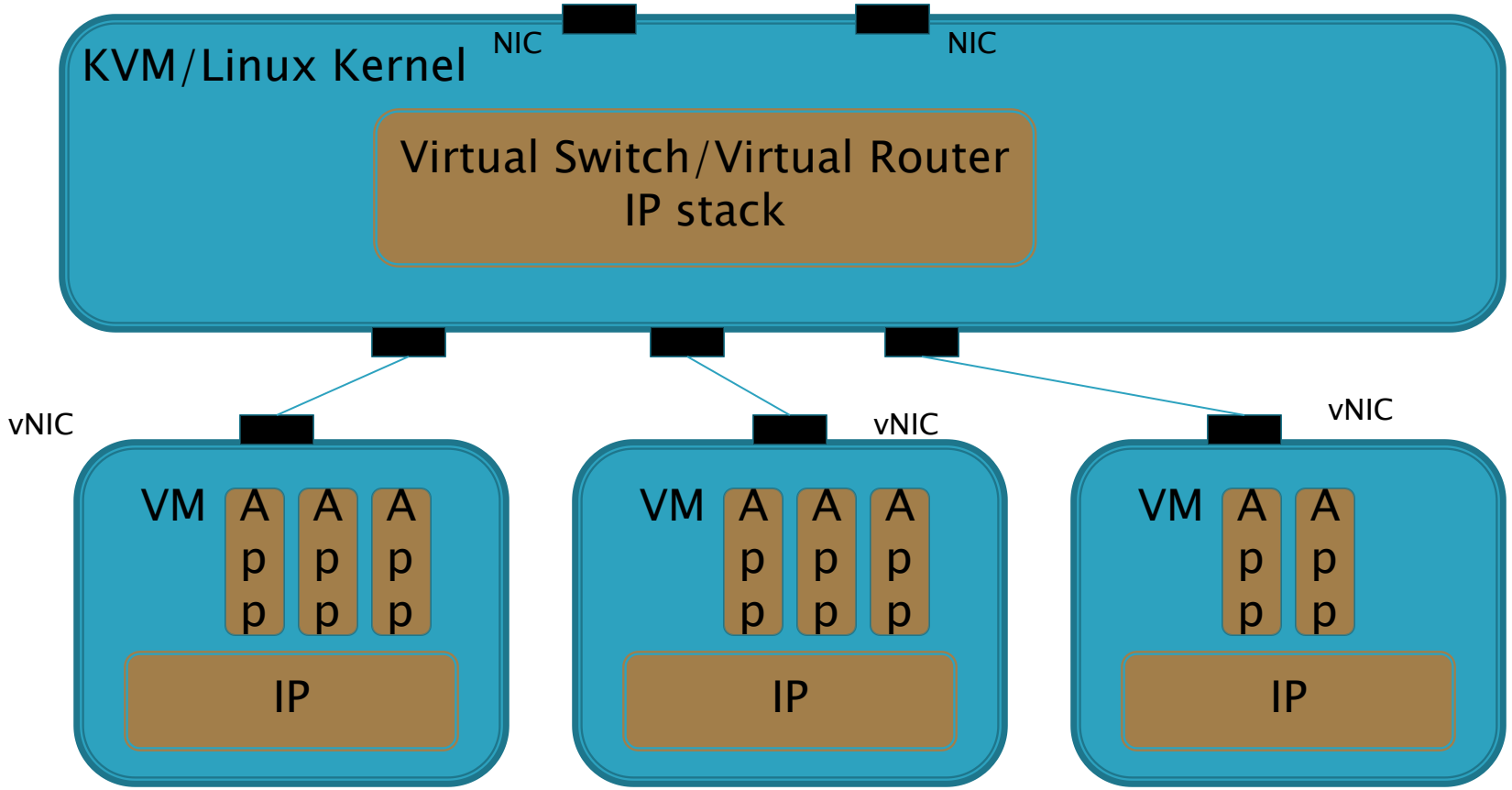
Bypass Linux Kernel Stack (Netmap)



Test result shows NETMAP with 60B packet costs ~1000cycles/packet

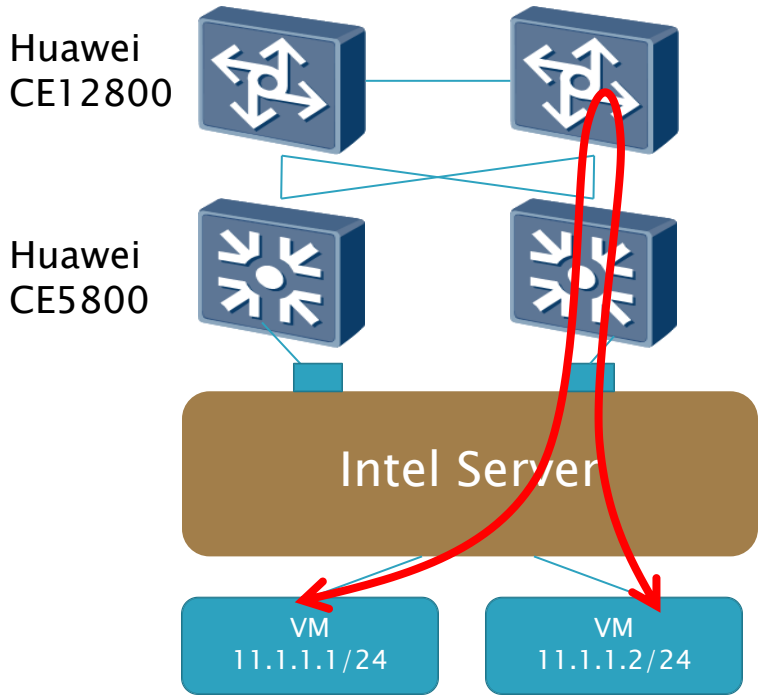
* Netmap is a open source at *http://info.iet.unipi.it/~luigi/netmap/*

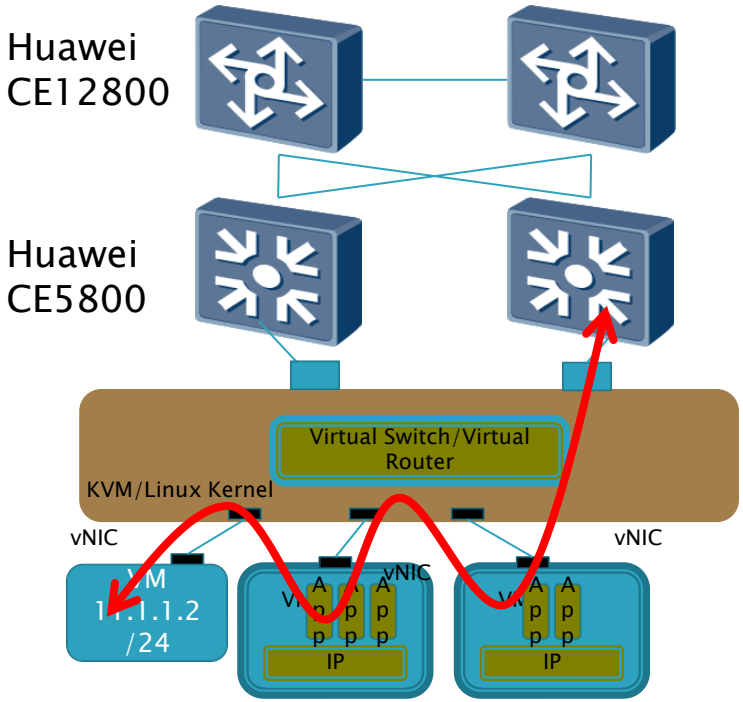
Issue Two: Two stacked IP stacks



SR-IOV is a possible solution, but yet introduces other problems

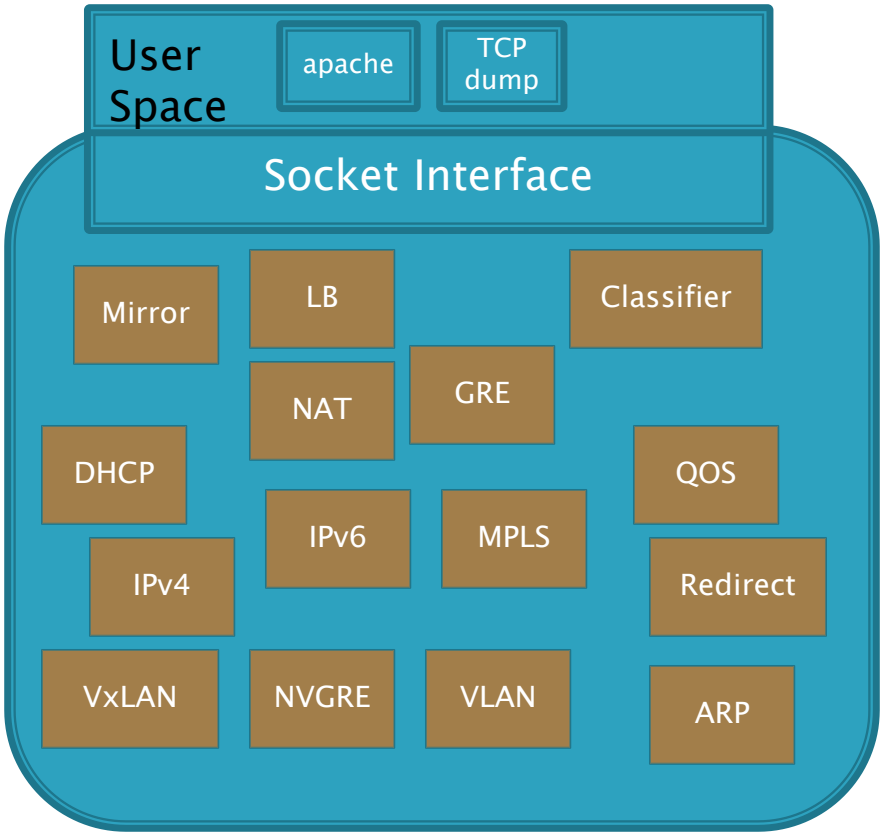
Issue Three: From Old Hairpin to New Hairpin





In NFV, VM may not be the endpoint anymore

Issue Four: Getting Crowded Kernel/Hypervisor

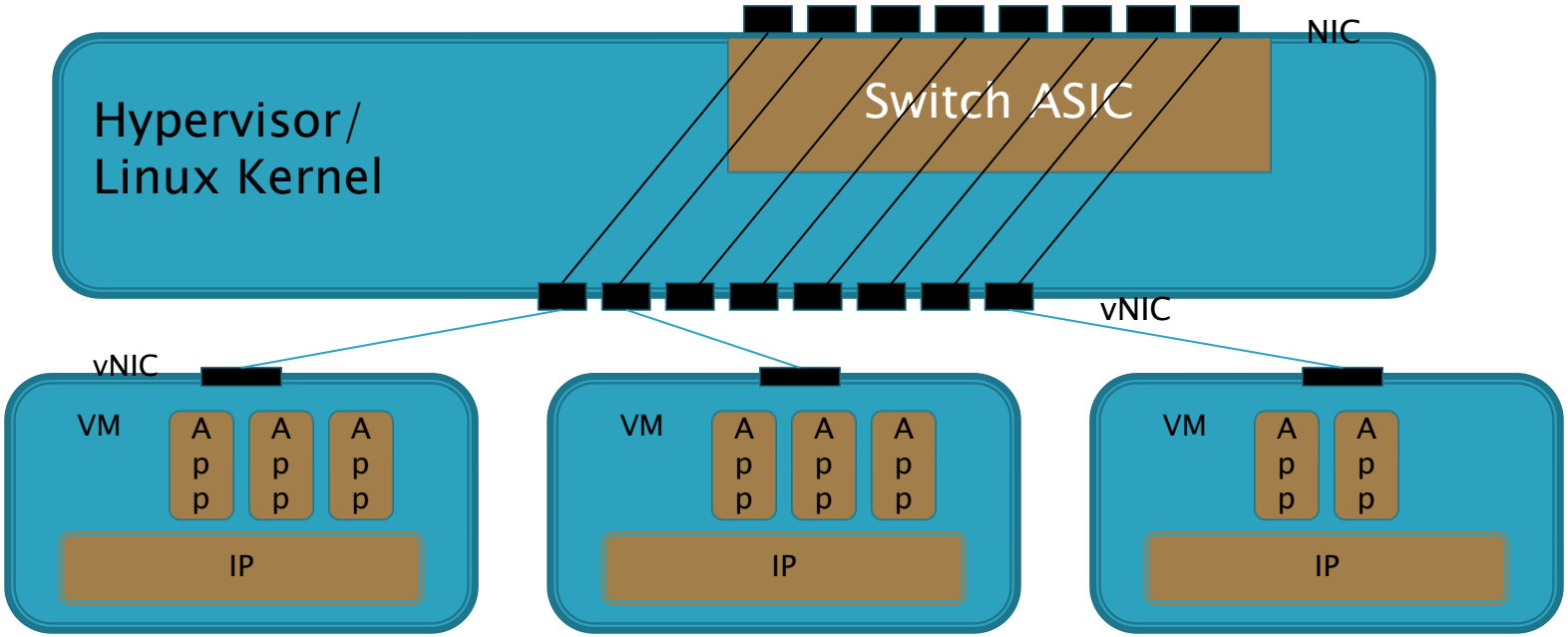


Add (Distributed) Routing
Add MPLS
Add (Distributed) Firewall
Add QOS
Add IP Filter
Add Packet Classifier
Add Redirect
Add Load Balance
Plus existing L2 functions
….
All into (*Linux*) Kernel

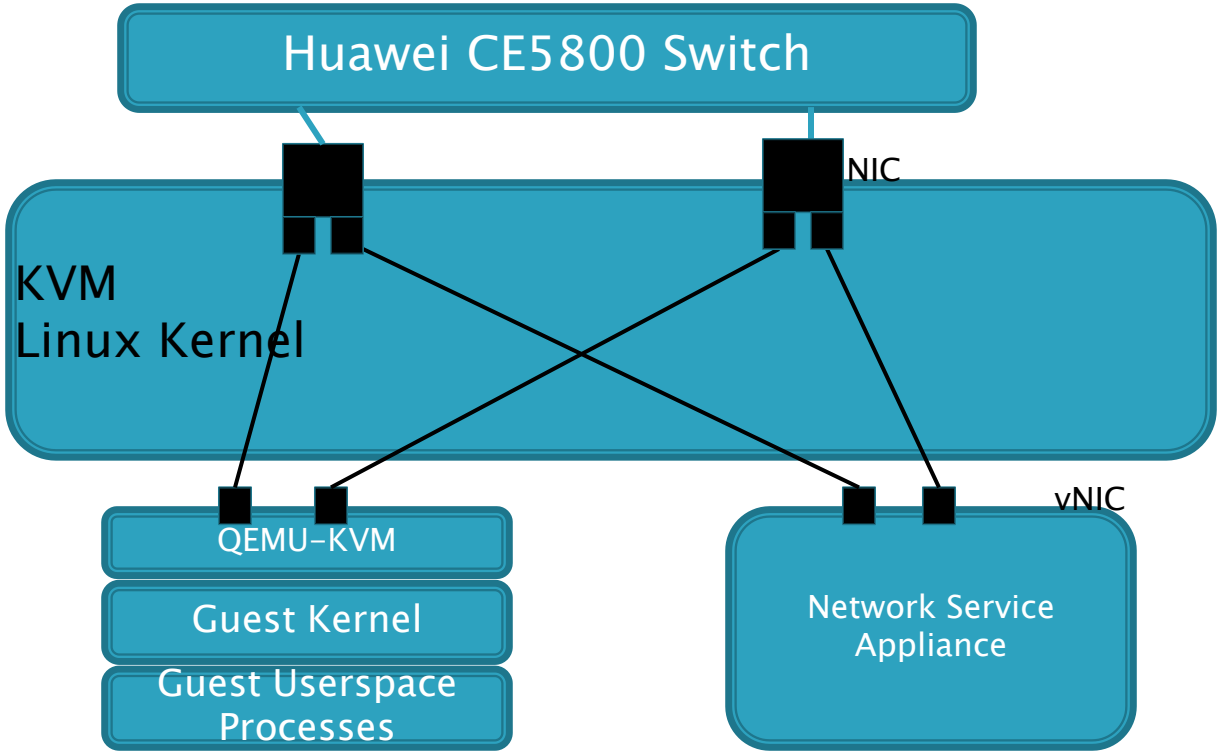
OS/hypervisor/Network be the monolithic piece of all (Sounds familiar?)

Learn from Existing Approaches

Use ASIC to offload the switching function
Common approach from Network Vendors
Challenges to address portability, feature velocity in ASICs



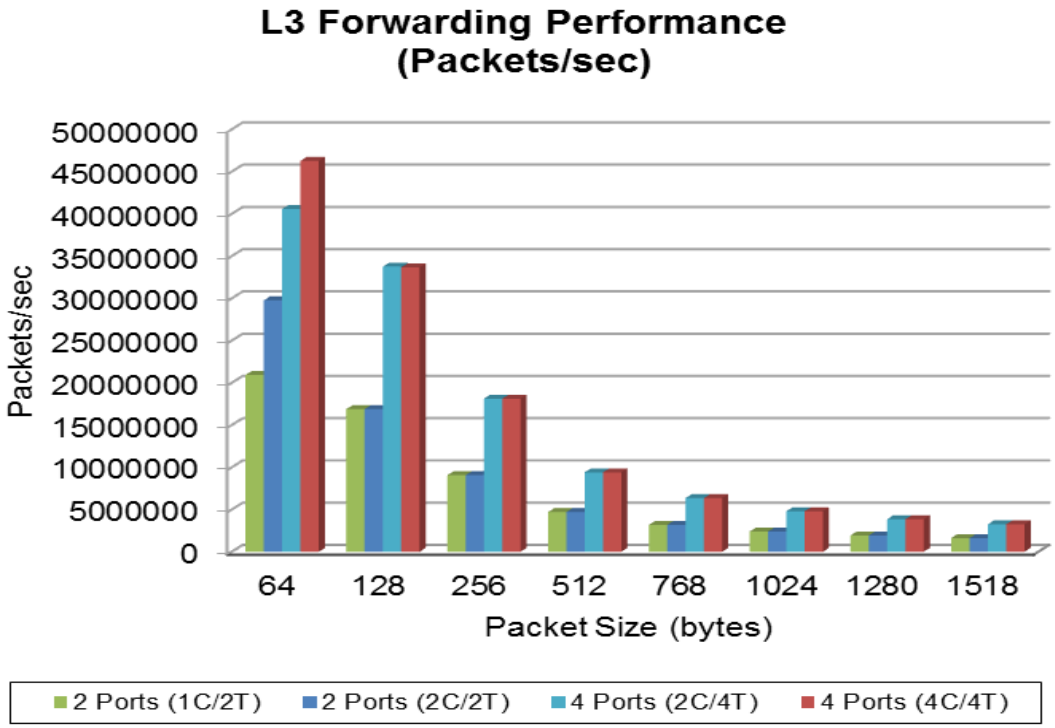
Can We Solve these Issues with Software?



- SR-IOV provides the separated access to a network adaptor among various PCIe hardware functions.
- It bypasses the virtual switch function in the kernel to allow network traffic directly goes between VF and VM
- Combine with UIO, QEMU can access the FV and provide the IO to Guest Kernel
- SR-IOV and UIO are orthogonal technology. Other IO solutions are also feasible.

DPDK for Network Services

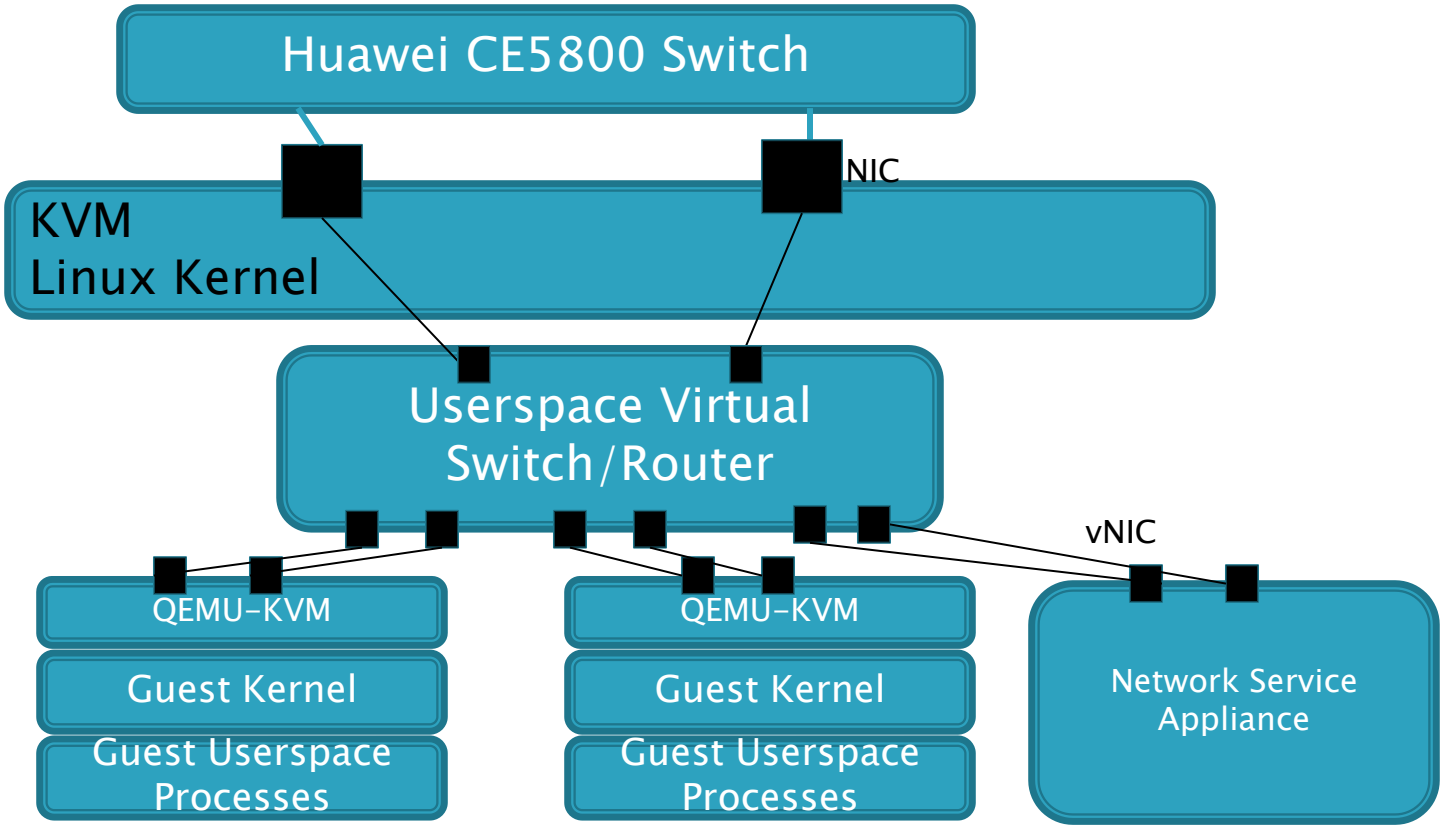
- DPDK demonstrates the desired network packet performance for NFV
- DPDK userspace design points to a better software approach
- Packet processing enhancement provides further opportunities:
  - Integration of High Bandwidth PCIe Gen3
  - New AVX Extensions
  - Intel® Virtualization Technology (Intel® VT)
  - Intel® Data Direct I/O Technology (Intel® DDIO)



- Quad Core Intel® Core™ i7-3610QE Processor 2.30GHz (E1), 6MB L3 cache
- Mobile Intel® QM77 Express Chipset (A1)
- Emerald Lake 2 Platform (CRB)
- DDR3 1600MHz, 2 x dual rank 4GB (total 8GB), Dual-Channel Configuration
- 2 x Intel® 82599 Dual Port PCI-Express x8
  10 Gigabit Ethernet NIC

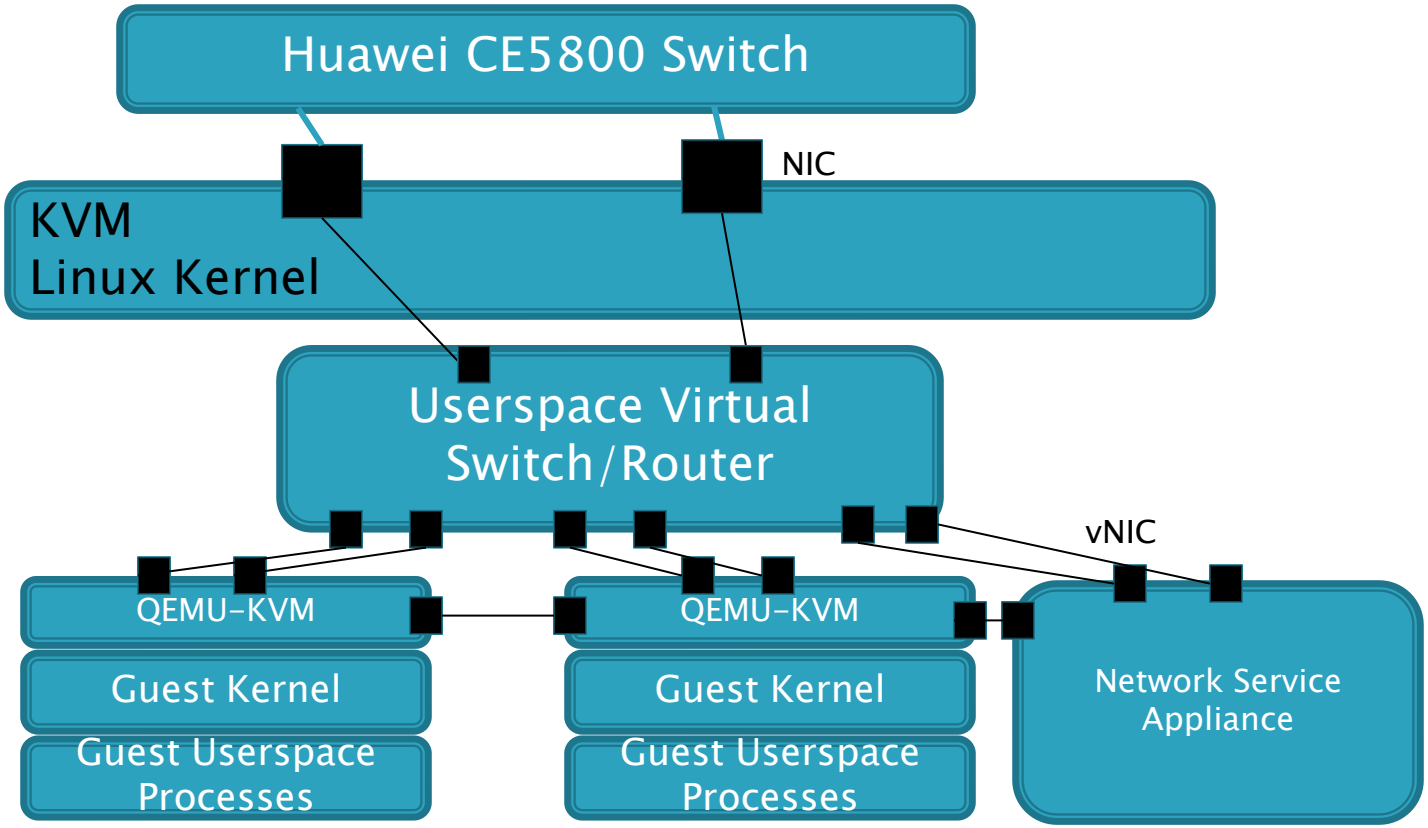
From Intel DPDK. For DPDK reference to *http://www.intel.com/go/dpdk

Userspace Virtual Switch/Router



- KVM/Linux Kernel back to what it is designed for, and yet robust.
- Userspace Virtual Switch/Router uses UIO to directly access Physical NIC, and benefits from DDIO, PCIe Gen3, IOTLB, etc
- Userspace Virtual Switch/Router provides vNIC for VM for accomplishing the inter-VM communication
- Elastic performance with multi-core support.

InterVM Direct Packet Path



- Userspace packet path can be directly from VM to VM
- Zero copy is possible for inter-VM packets in the same host when optimizing with Guest kernel UIO, and Frontend Driver in QEMU.

Future Work

- Support more IO types
- Less intrusive IO
- Feature Rich Userspace Switch/Router
- Multi-core expansion
- Multiple Instances to support Multi-tenancy

Summary

- Monolithic hypervisors may not be extensible with NFV deployment
- Linux/KVM/hypervisor can focus on its main tasks, e.g. process scheduling, resource management, virtualization
- Userspace IP stack provides another approach for easy development environment, high packet performance, and compatible VM communication support.

Reference and Contact

- Reference

  DPDK *https://01.org/packet-processing/overview/dpdk-detail*
  NETMAP *http://info.iet.unipi.it/~luigi/netmap/*
  PR_RING *http://www.ntop.org/products/pf_ring/*
  KVM *http://www.linux-kvm.org/page/Main_Page*

- Contact me at jun.xu@huawei.com