intel Look Inside.™

Additional DPDK Theory

Network Platforms Group

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm%20 Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Celeron, Intel, Intel logo, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel SpeedStep, Intel XScale, Itanium, Pentium, Pentium Inside, VTune, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel® Active Management Technology requires the platform to have an Intel® AMT-enabled chipset, network hardware and software, as well as connection with a power source and a corporate network connection. With regard to notebooks, Intel AMT may not be available or certain capabilities may be limited over a host OS-based VPN or when connecting wirelessly, on battery power, sleeping, hibernating or powered off. For more information, see http://www.intel.com/technology/iamt.

64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

No computer system can provide absolute security under all conditions. Intel® Trusted Execution Technology is a security technology under development by Intel and requires for operation a computer system with Intel® Virtualization Technology, an Intel Trusted Execution Technology-enabled processor, chipset, BIOS, Authenticated Code Modules, and an Intel or other compatible measured virtual machine monitor. In addition, Intel Trusted Execution Technology requires the system to contain a TPMv1.2 as defined by the Trusted Computing Group and specific software for some uses. See http://www.intel.com/technology/security/ for more information.

†Hyper-Threading Technology (HT Technology) requires a computer system with an Intel® Pentium® 4 Processor supporting HT Technology and an HT Technology-enabled chipset, BIOS, and operating system. Performance will vary depending on the specific hardware and software you use. See www.intel.com/products/ht/hyperthreading_more.htm for more information including details on which processors support HT Technology.

Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, virtual machine monitor (VMM) and, for some uses, certain platform software enabled for it. Functionality, performance or other benefits will vary depending on hardware and software configurations and may require a BIOS update. Software applications may not be compatible with all operating systems. Please check with your application vendor.

* Other names and brands may be claimed as the property of others.

Other vendors are listed by Intel as a convenience to Intel's general customer base, but Intel does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices. This list and/or these devices may be subject to change without notice.

Copyright © 2015, Intel Corporation. All rights reserved.

Topics

- DPDK Libraries introduction
- DPDK Sample Applications
- Load Balancing, packet scheduler libraries, packet distributor
- Packet generating tools
- DPDK virtualisation

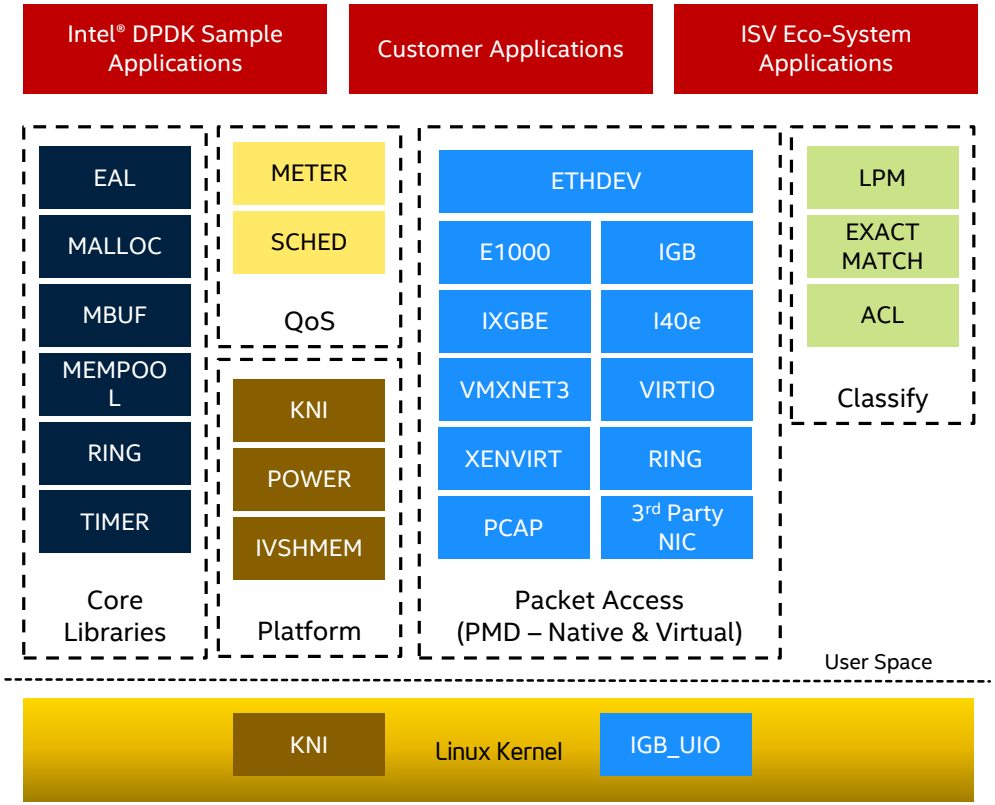
DPDK Libraries

What is DPDK?

Short Answer:

The DPDK is a set of software libraries designed for high-speed packet processing

Data Plane Development Kit



- Libraries for network application development on Intel Platforms
  - Speeds up networking functions
  - Enables user space application development
  - Facilitates both run-to-completion and pipeline models
- Free, Open-sourced, BSD Licensed
  - Git: http://dpdk.org/git/dpdk
- Scales from Intel Atom to multi-socket Intel Xeon architecture platforms and other architectures
- About two dozen pre-built example applications

DPDK Libraries and Drivers

Poll Mode Drivers: DPDK includes Poll Mode Drivers for multiple Ethernet controllers which are designed to work without asynchronous, interrupt-based signaling mechanisms, which greatly speeds up the packet pipeline. Different Virtual interfaces supported as well – virtio, user space vHost, pcap, shared memory, etc.

Memory Manager: Responsible for allocating objects in memory. Can provide memory blocks and object pools in huge page memory. It also provides an alignment helper to ensure that objects are padded to spread them equally on all DRAM channels.

Buffer Manager: Reduces by a significant amount the time the operating system spends allocating and de-allocating buffers. DPDK pre-allocates fixed size buffers which are stored in memory pools.

Queue Manager: Implements safe lockless queues, instead of using spinlocks, that allow different software components to process packets, while avoiding unnecessary wait times.

Flow Classification: Provides an efficient mechanism to produce a hash based on tuple information so that packets may be placed into flows quickly for processing, thus greatly improving throughput.

The libraries/components (1)

Library	
librte_eal	Environment Abstraction Layer. Meant to hide system/OS specifics from “common” upper layers
librte_malloc	rte_malloc() - replacement for malloc(). Allows allocation of data structures backed by huge pages
librte_mempool librte_mbuf	Memory management: DPDK buffer pool management and packet buffer implementations
librte_ring	High speed ring for inter-core/process pointer passing
librte_timer	Timer routines
librte_lpm	Accelerated longest prefix match
librte_hash	Hash driven key-value exact match for tuple matching
librte_acl	Accelerated implementation of an Access Control List

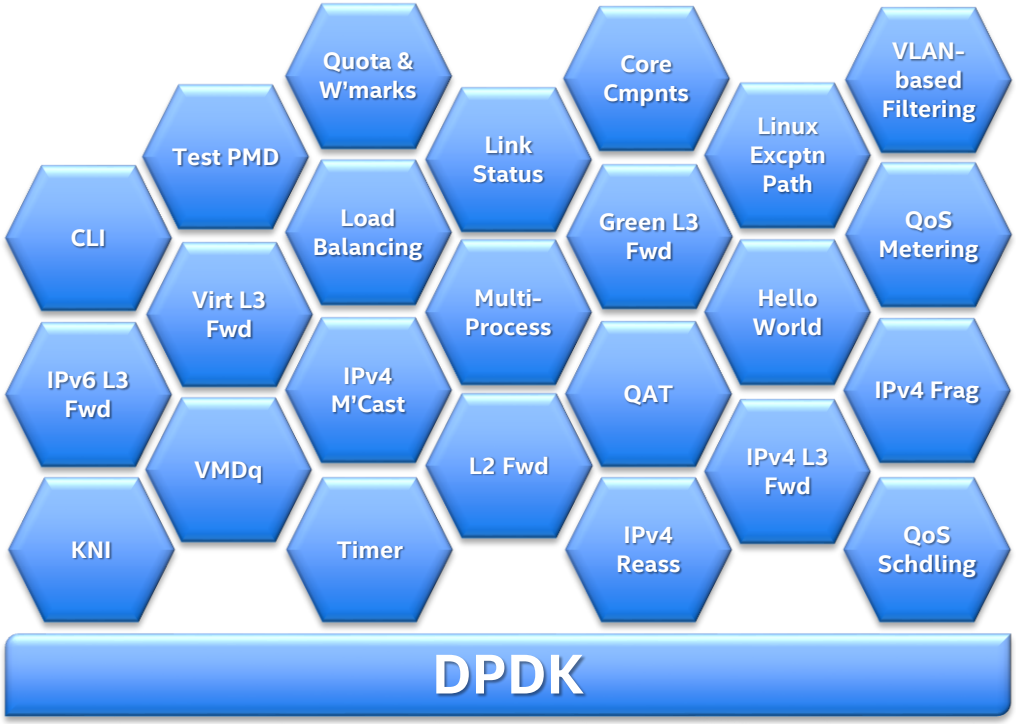
The libraries/components (2)

Library	
librte_meter	Meter/mark library: Implements srTCM (RFC 2697) and trTCM RFC 2698)
librte_sched	Hierarchical traffic shaper in software
librte_pmd*	Packet Access “Poll” mode drivers
librte_ether	Generic Ethernet device abstraction – the DPDK PMD API
librte_cmdline	Command line parser library
librte_distributor	A work queue distributor
librte_power	Power management primitives
librte_ivshmem	Shared memory implementation for inter-VM communication
KNI, librte_kni	Kernel Network Interface – implements a kernel netdev for passing packets into the kernel from DPDK

DPDK Sample Applications

Build with Intel® DPDK

Provided Sample Applications



- About two-dozen pre-built sample applications
- Provide a great jump start for accelerating workloads with DPDK

Build with Intel® DPDK

Provided Sample Applications

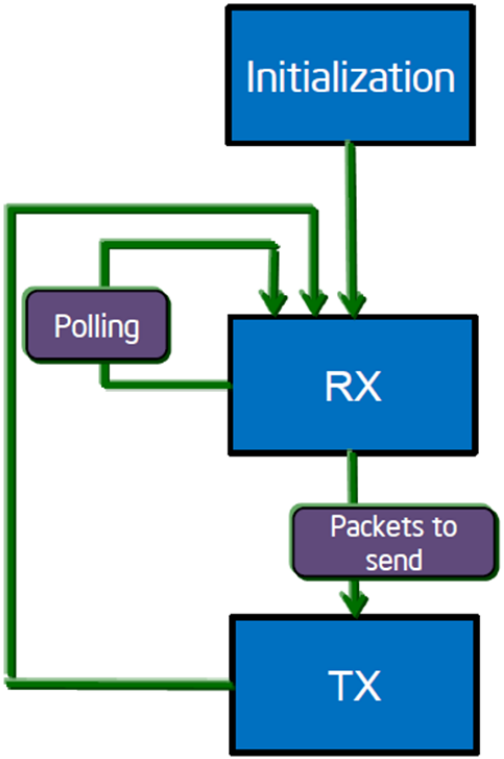
- Test PMD application
  - Support for a variety of PMD driver cases
  - Support for Flow Director provided in the Intel® 82599 10 Gigabit Ethernet Controller
- Test application
  - Support for core component tests
- Sample applications
  - Command Line
  - Exception Path (into Linux* for packets using the Linux TUN/TAP driver)
  - Hello World
  - Integration with Intel® Quick Assist Technology drivers 1.0.0 and 1.0.1 on Intel® Communications Chipset 89xx Series C0 and C1 silicon.
  - Link Status Interrupt (Ethernet* Link Status Detection
  - IPv4 Fragmentation
  - IPv4 Multicast
  - IPv4 Reassembly
  - L2 Forwarding (supports virtualized and non-virtualized environments)

Build with Intel® DPDK

Provided Sample Applications

- L3 Forwarding (IPv4 and IPv6)
- L3 Forwarding in a Virtualized Environment
- L3 Forwarding with Power Management
- QoS Scheduling
- QoS Metering + Dropper
- Quota & Watermarks
- Load Balancing
- Multi-process
- Timer
- VMDQ and DCB L2 Forwarding
- Support VMDq for 1 GbE and 10 GbE NICs to demonstrate VLAN-based packet filtering
- Kernel NIC Interface (with ethtool support)
- Multi-process example using fork() to demonstrate application resiliency and recovery, including reattachment to and re-initialization of shared data structures where necessary

DPDK Skeleton App: basefwd



Initialization

- Initialize memory zones and pools
- Initialize devices and device queues
- Start the packet forwarding application

Packet Reception (RX)

- Poll devices' RX queues and receive packets in bursts
- Allocate new RX buffers from per queue memory pools to stuff into descriptors

Packet Transmission (TX)

- Transmit the received packets from RX
- Free the buffers used to store the packets to send

DPDK Skeleton APP pseudo code

```
int main(int argc, char **argv)
{
       rte_eal_init(argc, argv); // initialize EAL
       rte_mempool_create(...); // buffers preallocation (NUMA aware)
   ... // initialize ports
       rte_eth_dev_configure(port_id, 1, 1, &port_conf);
       rte_eth_rx_queue_setup(port_id, queue, ...);
       rte_eth_tx_queue_setup(port_id, queue, ...);
   ...
       rte_eth_dev_start(portid);
       rte_eth_promiscuous_enable(portid);
       for(;;) {
          nb_rx = rte_eth_rx_burst(portid, queue,packets, MAX_PKT_BURST); // poll port
          ...
          rte_eth_tx_burst(portid, queue,  packets, nb_rx); // send to port
       }
}
```

Packet Distributor Sample Application

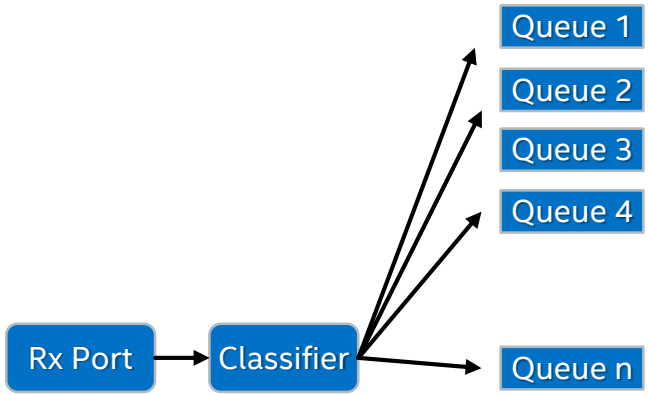
Problem Definition

❑ Packet Distribution

How to distribute packet flows across queues

Packet Distributor

- Directing flows to queues

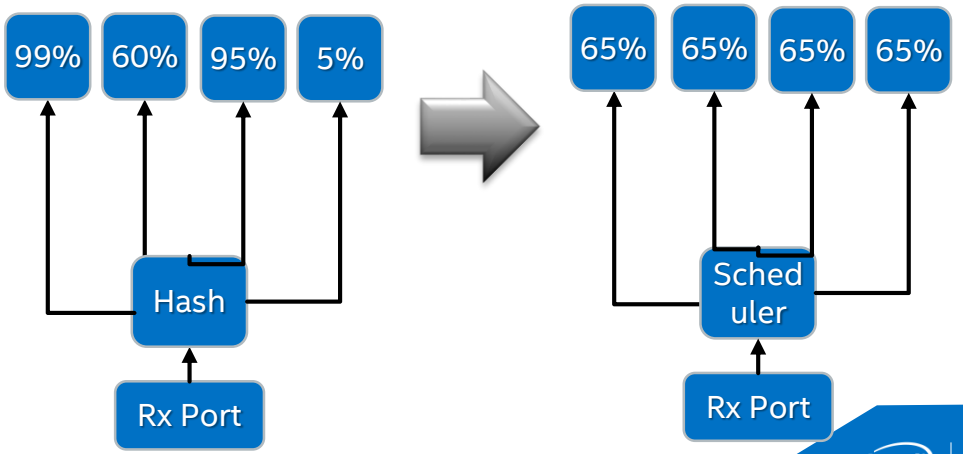


❑ Load Balancing

How achieve an even balance of load on each core.

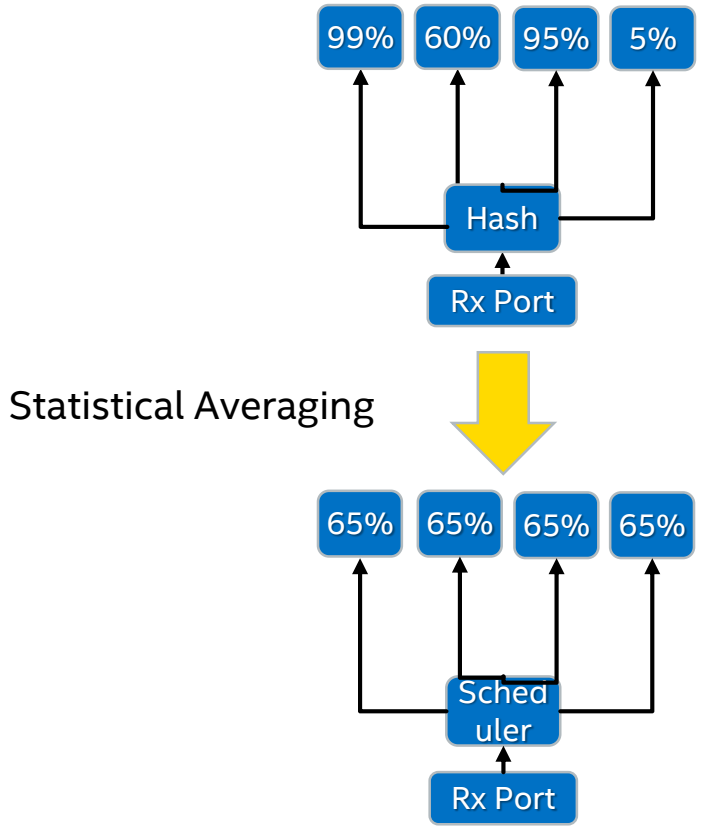
Load Balancing

- Even distribution of workload across cores



DPDK In- order distributor achieves balancing

- Manage 10G
- Per Packet processing
- Load Aware
- Run to complete
- Flow affinity
- In order
- Sample App
- Any Hash
- TX coalescing
- QoS **- N**



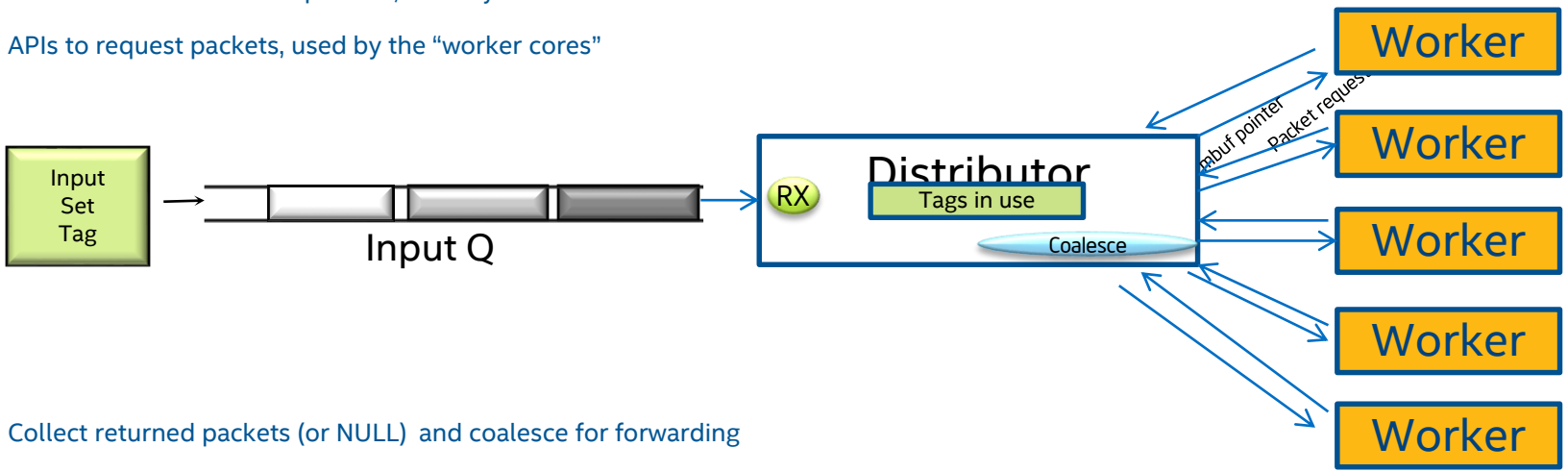
In Order Packet Distributor

New Intel® DPDK component usable in packet pipeline

Works with each packet individually

APIs to distribute a set of packets, used by the “distributor core”

APIs to request packets, used by the “worker cores”



Collect returned packets (or NULL) and coalesce for forwarding

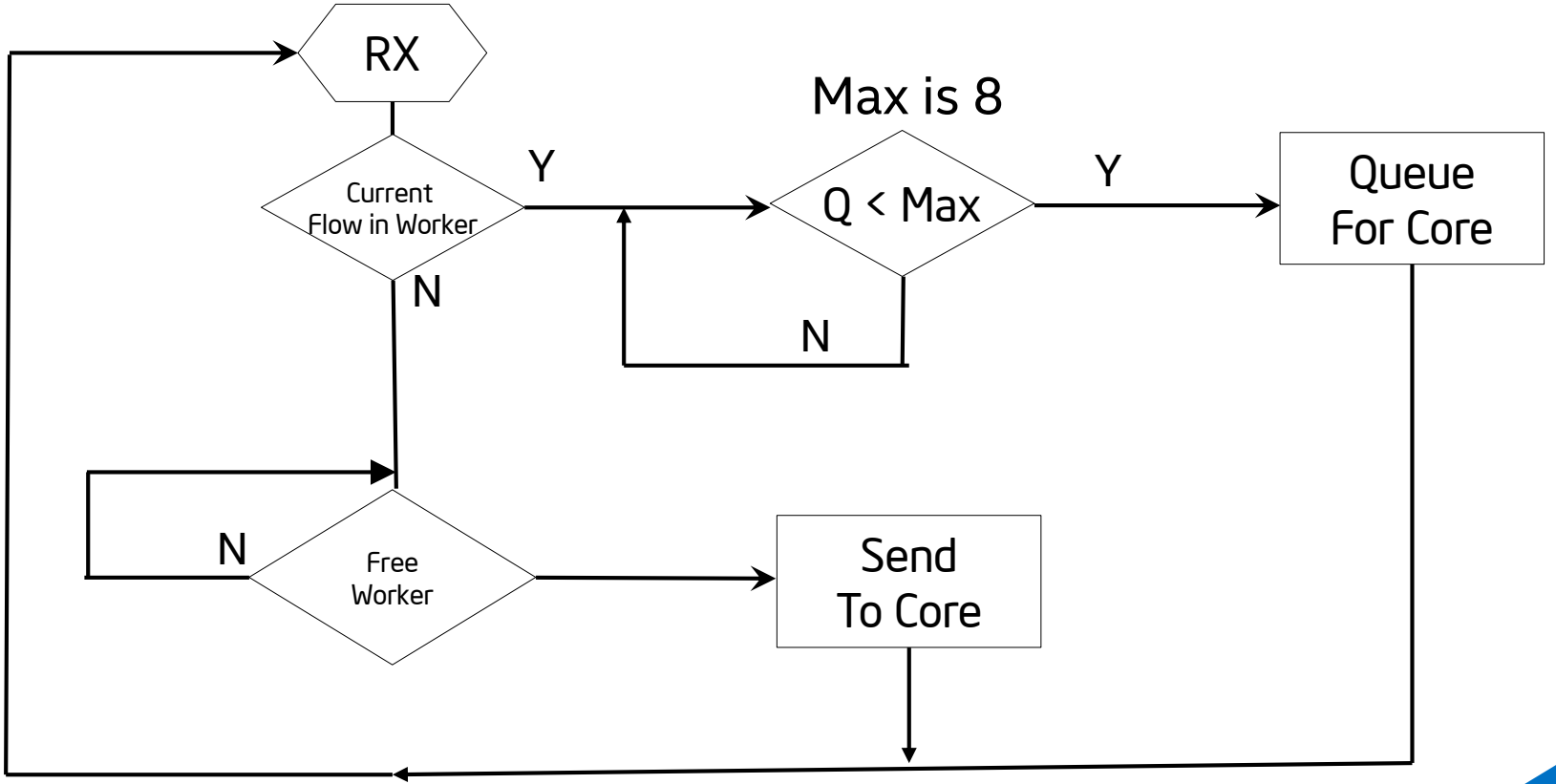
In order processing within a flow, identified by a tag

DPDK run to complete target architecture

Statistical balancing over N flows >> Workers

TAG can be any Hash, Sample APP uses NIC RSS for example

Distributor Logic



DPDK Packet Re-Order

DPDK 2.0

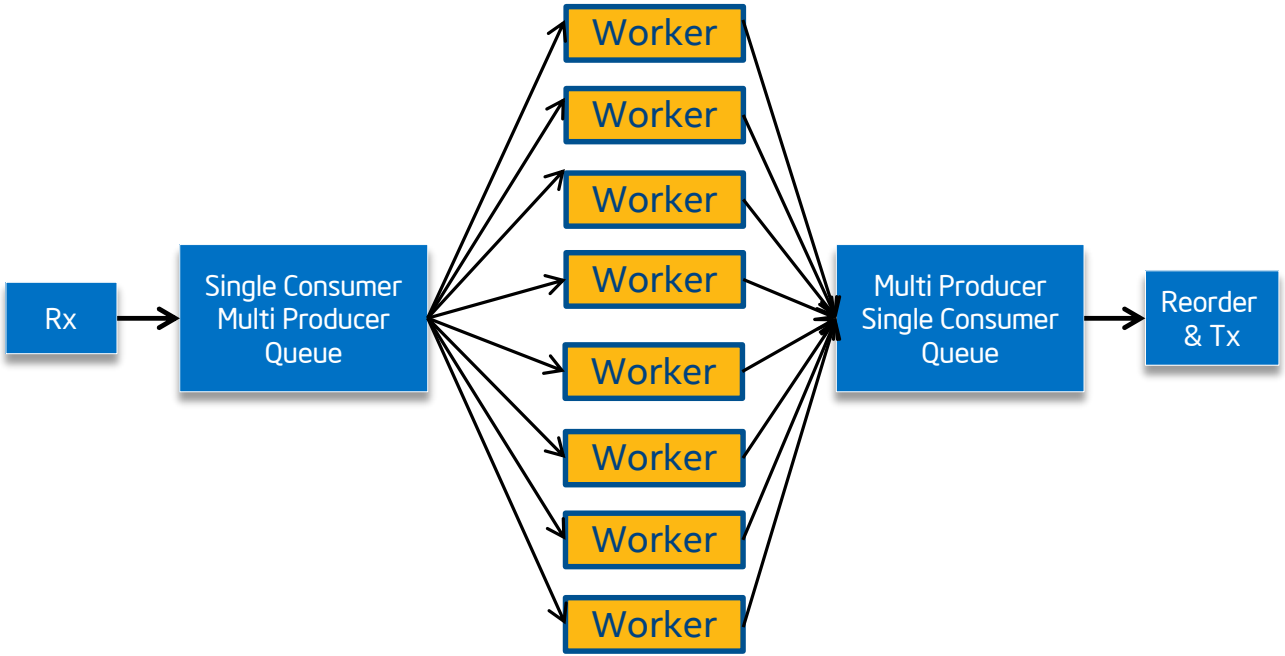
DPDK 2.0 – Packet Re-ordering Feature

DPDK Packet Distributor unchanged since 1.7

Packet Re-ordering library and sample app new to 2.0

- Standalone sample application
- Out of order packet processing on multiple cores
- Improved performance on Packet Distributor application

DPDK Packet Re-order



DPDK Packet Re-order

Still a prototype – further investigation in progress

- Need to further optimise

No Hash value (e.g. RSS) used here (unlike Packet Distributor).

Flows are mixed – absolute sequence # for reordering

Prioritisation – not implemented in sample app

- Could implement with multiple queues on ingress side

DPDK Packet Re-order

Library – reorder block

- API – insert packet, drain packet

Currently, only 1 reorder buffer

- Multiple buffers require more complex logic
- It could be feasible, but not implemented (or prototyped) at this point.
- Would be application responsibility – no plans to put in library

Load Distribution Summary

Packet Distributor component

Can achieve balancing due to statistical behaviour of traffic

Single packet at a time, run to complete model

Supports flow affinity

Packet distributor has a set of defined behaviors, usage model by workers can vary

DPDK 2.0 allow random allocation of packets and re-ordering using Packet Re-order sample app.

Packet Re-order works on batches of packets.

DPDK-based Packet Generators

Generate traffic without need for dedicated HW

SW Packet Generators

Linux:

- **pktgen**: Linux packet generator is a tool to generate packets at high speed in the kernel (limited by Linux kernel capability)
- **Netperf**: benchmarkng tool for measuring networking performance (UDP/TCP)

DPDK based:

- **MoonGen**: fully scriptable high-speed packet generator built on DPDK and LuaJIT https://github.com/emmericp/MoonGen
- **Oscinato:** https://code.google.com/p/ostinato/
  DPDK accelerated: http://www.slideshare.net/pstavirs/dpdk-accelerated-ostinato
- **DPPD v017**: https://01.org/intel-data-plane-performance-demonstrators/downloads
- **pktgen-dpdk**: traffic generator http://dpdk.org/browse/apps/pktgen-dpdk

Pktgen-dpdk

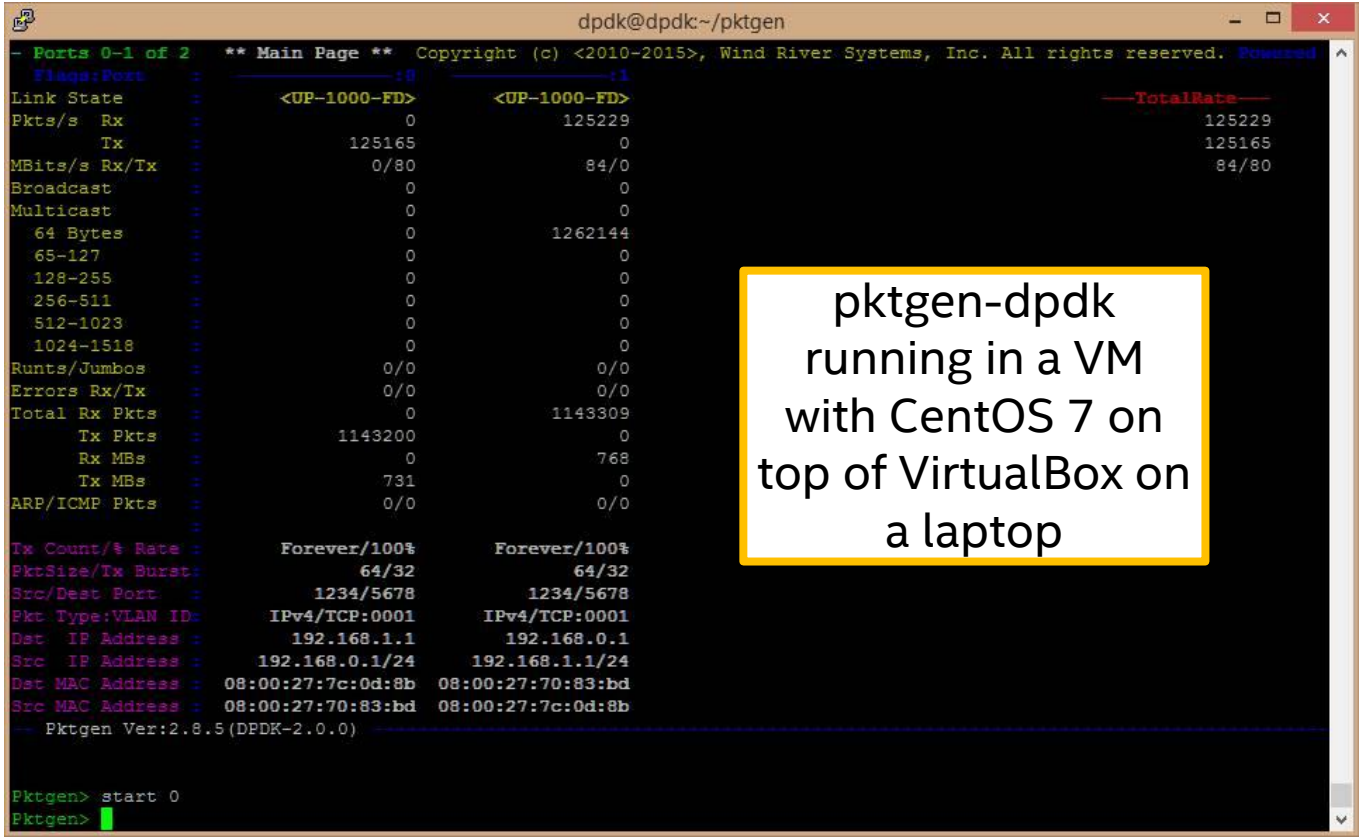
- It is capable of generating 10Gbit wire rate traffic with 64 byte frames.
- It can act as a transmitter or receiver at line rate.
- It has a runtime environment to configure, and start and stop traffic flows.
- It can display real time metrics for a number of ports.
- It can generate packets in sequence by iterating source or destination MAC, IP addresses or ports.
- It can handle packets with UDP, TCP, ARP, ICMP, GRE, MPLS and Queue-in-Queue.
- It can be controlled remotely over a TCP connection.
- It is configurable via Lua and can run command scripts to set up repeatable test cases.
- The software is fully available under a BSD licence.

Pktgen-dpdk (continued)

- Pktgen-dpdk is freely available on www.dpdk.org
- Instructions are provided with the package
- Just a DPDK application
- Can be used on virtual or physical network devices – like DPDK
- More info at http://pktgen.readthedocs.org/en/latest/

Used in the hands on sessions in this course

Pktgen-dpdk (continued)



pktgen-dpdk running in a VM with CentOS 7 on top of VirtualBox on a laptop

By default pktgen if configured with two ports will send packets from port 0 to port 1 – easy way to test performance

DPDK - Virtualisation

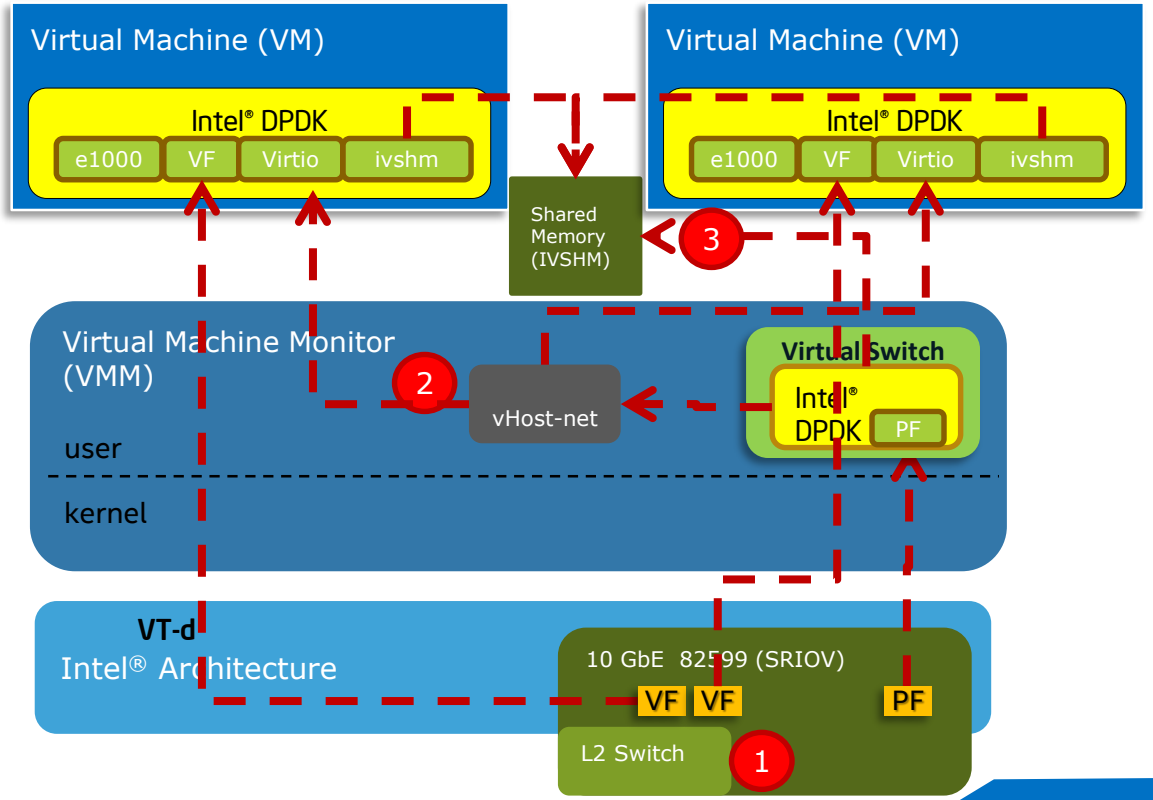
Intel® DPDK Virtualization Architecture

NFV requires fast VM-to-VM communication:

1. L2 Switching – Support Traffic Mirroring (Pool, VLAN, Uplink and Downlink) between VMs using Niantic

- Virtio PMD with Userspace
2. vhost back-end

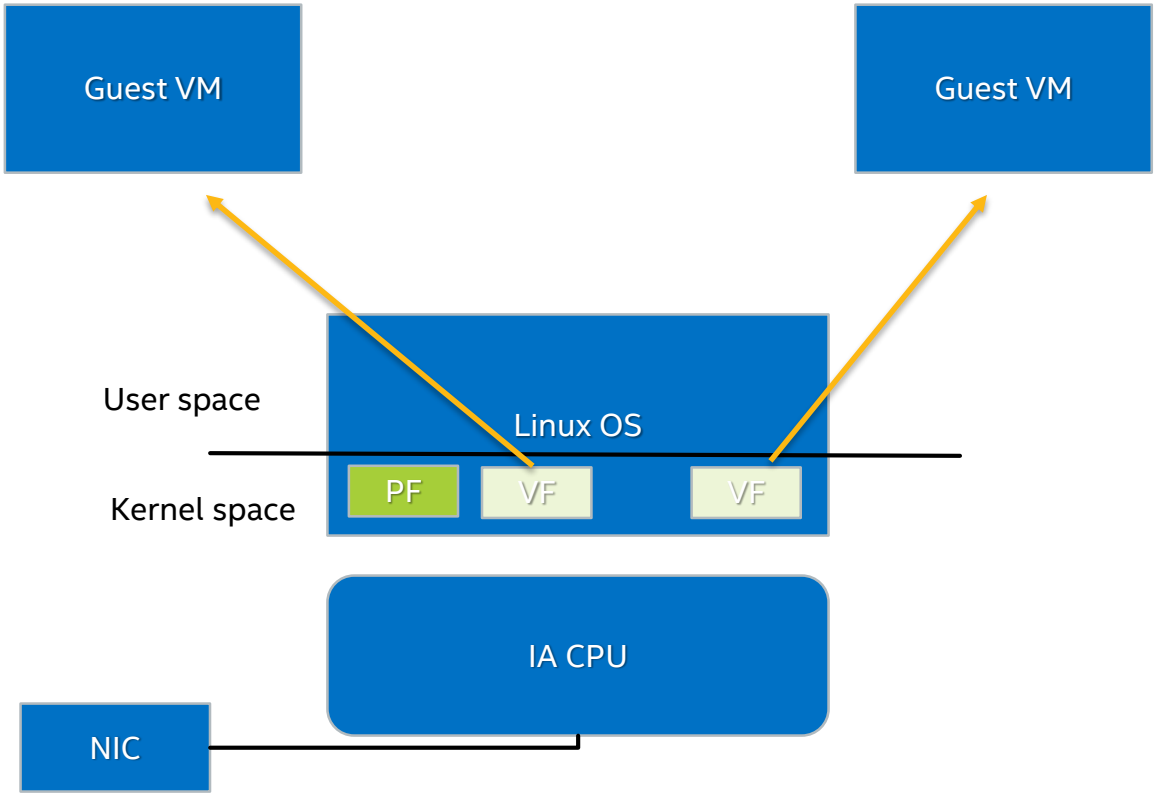
3. IVSHM support



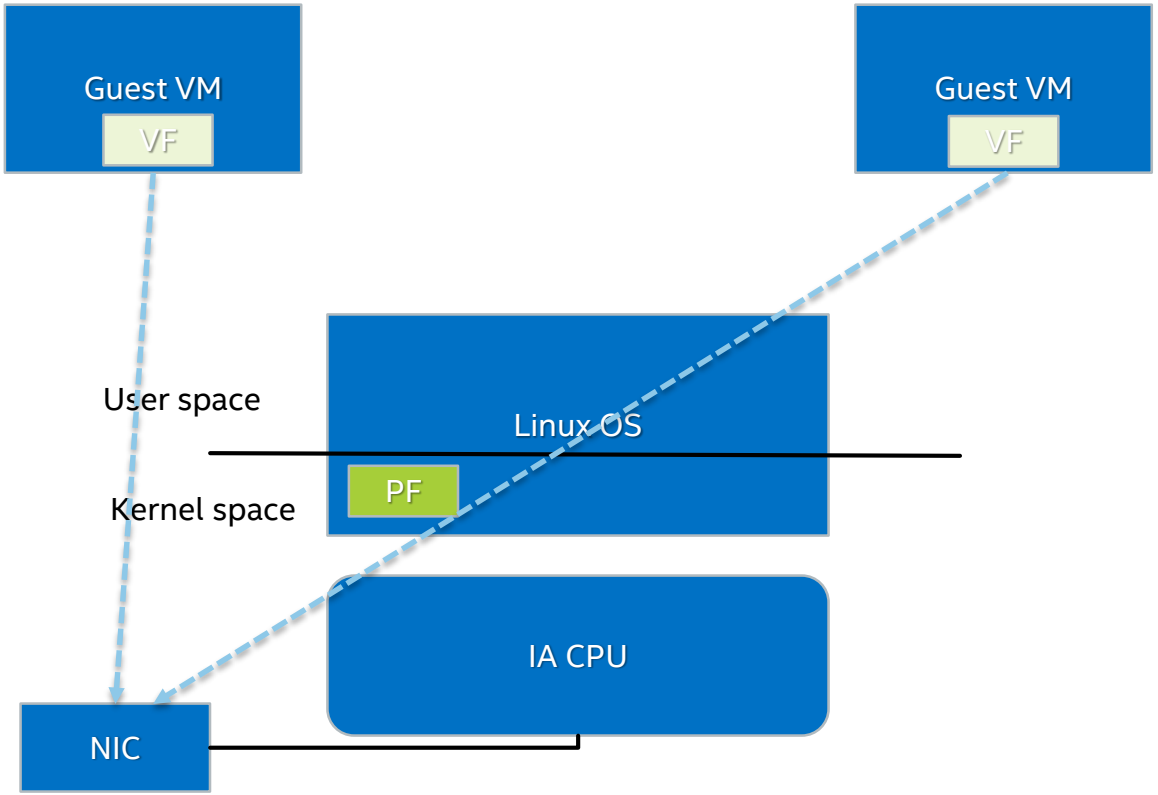
SR-IOV

- Single Root I/O Virtualisation
- Enables sharing of PCI resource between multiple VMs
- NIC will have Physical Function driver (PF)
- Also, can create multiple Virtual Function drivers (VF)
  - Can use PCI pass through to send VFs to guest VMs
  - PF can remain on host machine
  - VF driver effectively “bypasses” hypervisor
  - DPDK supports both PF and VF drivers
- Requires VT-d support
  - Be sure to enable in BIOS

SR-IOV (continued)



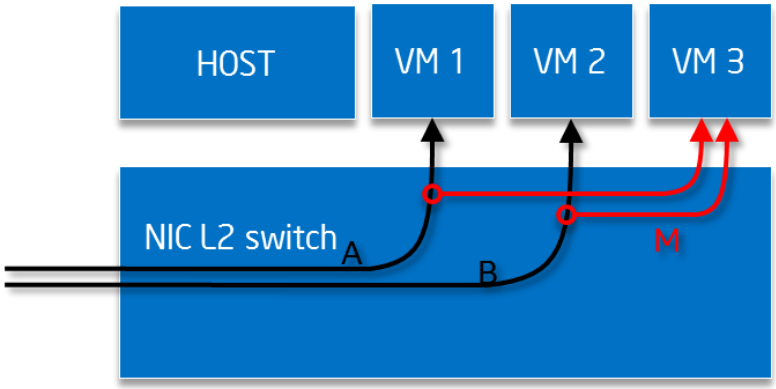
SR-IOV (continued)



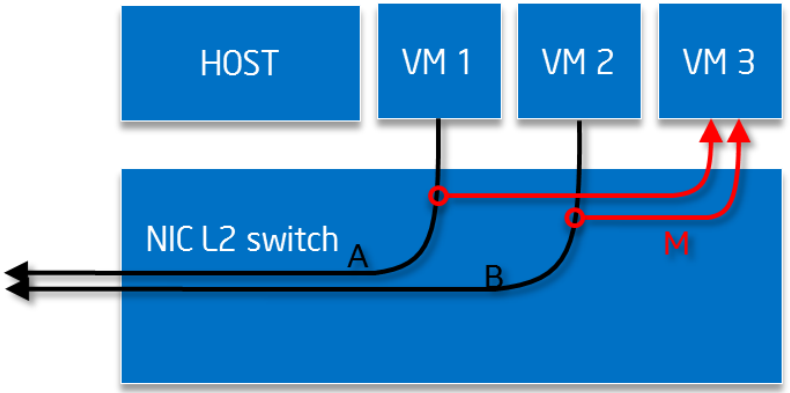
SR-IOV with DPDK

- VF passed through to guest VM can be Linux driver
  - Same limitations as Linux PF driver
- DPDK has VF driver also
  - Same advantages as DPDK PF driver
  - DPDK app in guest uses DPDK VF driver
  - Try with L2fwd sample application

SR-IOV mirroring

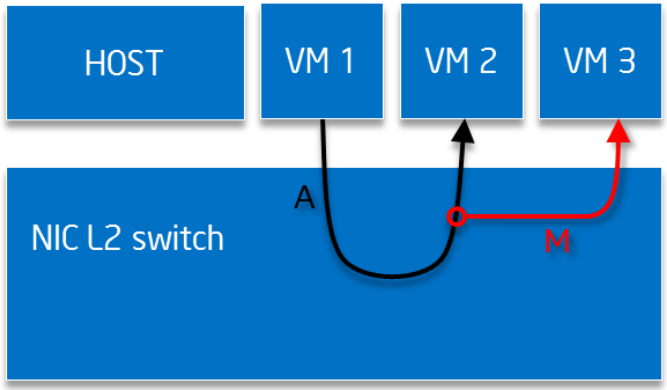


Uplink Mirroring



Downlink Mirroring

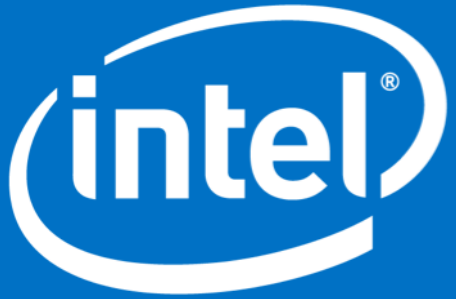
VF-toVF Mirroring



Note:
Niantic's L2 switch is limited to10GbE, including mirrored traffic

Vhost interface

- vhost-net Linux KVM para-virtualised interface on the guest VM
- vhost is the backend to this interface on the host
  - Performance was slow – vhost is in Linux kernel
  - Context switch between user and kernel space
- vhost userspace implementation
- Combine this with DPDK virtio PMD in guest for best performance
- See DPDK documentation and DPDK VHOST sample application for more information


intel