Network Interface Card

17TH FEB, 2020

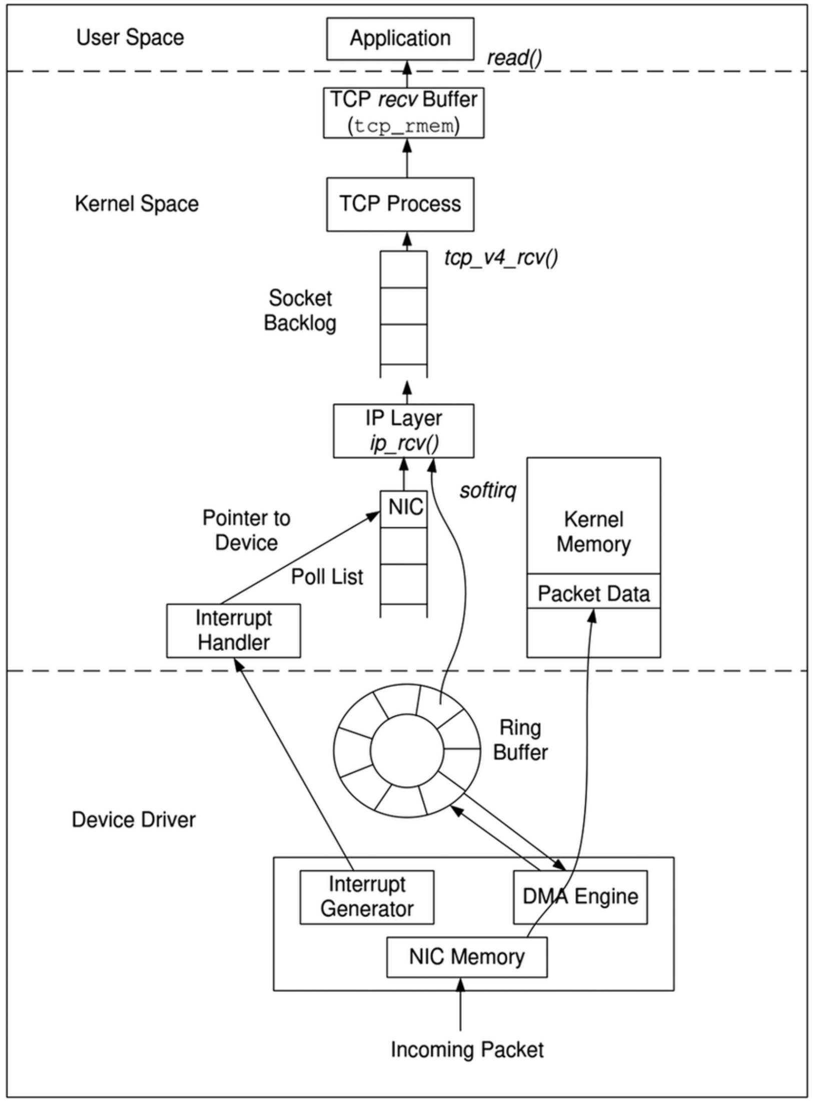
SAHIL SEMICONDUCTOR

NIC (Network Interface Card)

- NIC provides a way for a computer to connect to a network.
- Each NIC port has a unique 48-bit MAC address.
- Newer NICs which support SR-IOV have a MAC address for each PF and VF.
- NICs provide a range of interface speeds:
  1. 1 Gbps:
     - cat5 ethernet cable
  2. 10 Gbps:
     - XFP (10 gigabit small form factor pluggable) or SFP+ (enhanced small form-factor pluggable)
  3. 25 Gbps:
     - SFP28 connector. Identical in dimensions to SFP+ connector.
  4. 100Gbps:
     - QSFP28 (Quad Small Form-factor Pluggable)

Typical Packet Flow (NIC + Host Stack)

- NIC DMAs an incoming packet to the received ring buffer in host memory.
- It generates an interrupt notifying the host that a packet is available to be picked up.
- Linux host executes an ISR (Interrupt Service Routine).
- ISR schedules a SoftIRQ (SoftIRQ is a software interrupt handler. They run at higher priority than any user space thread).
- Once SoftIRQ starts executing, it fetches the packet from the ring buffer and pushes it to the Linux networking stack using skb structures.



Packet Reception with NAPI

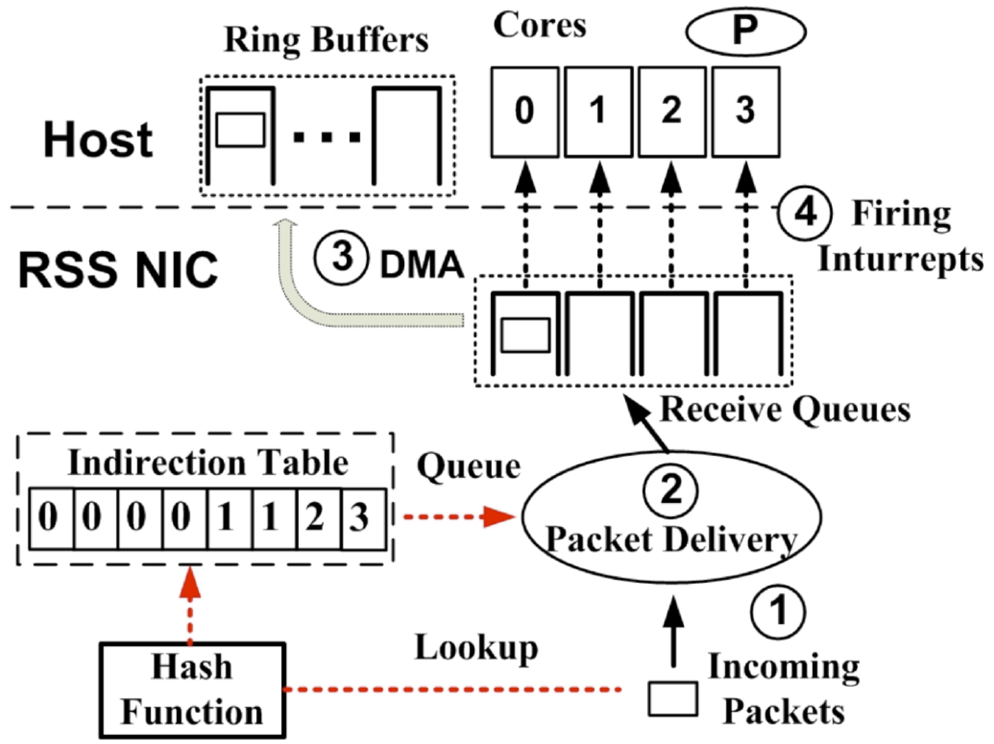
- Originally, Linux used to take one interrupt for each received packet.
  - This could cause excessive overhead under heavy load conditions.
- With NAPI (New API), the ISR notifies SoftIRQ that packet are available. It also disables incoming interrupts.
- SoftIRQ now polls for incoming packets (at-most a pre-defined packet count) and pushes them to the networking stack.
- Once done (either processed pre-defined number of packets, or there are no more packets available), it enables interrupts.
- NAPI avoids interrupt overhead for each packets.
- Most of the Linux NIC drivers have NAPI support.

Dumb NICs are Getting Smarter

- Historically, NIC only performed packet send/receive functionality.
- Over the period of time, they are getting more intelligent, supporting a number of offloads.
- Some of these features include:
  - TCP/UDP checksum offload
    - Performs CPU intensive TCP/UDP checksum calculation and verification in hardware.
  - Interrupt Coalescing
    - Do not generate interrupt for each incoming packet.
    - Only notifies the host through an interrupt when certain number of packets have been received or a timeout occurrs.
  - VLAN filtering and tag stripping:
    - Strip 802.1Q header and store VLAN ID in network packet meta data.

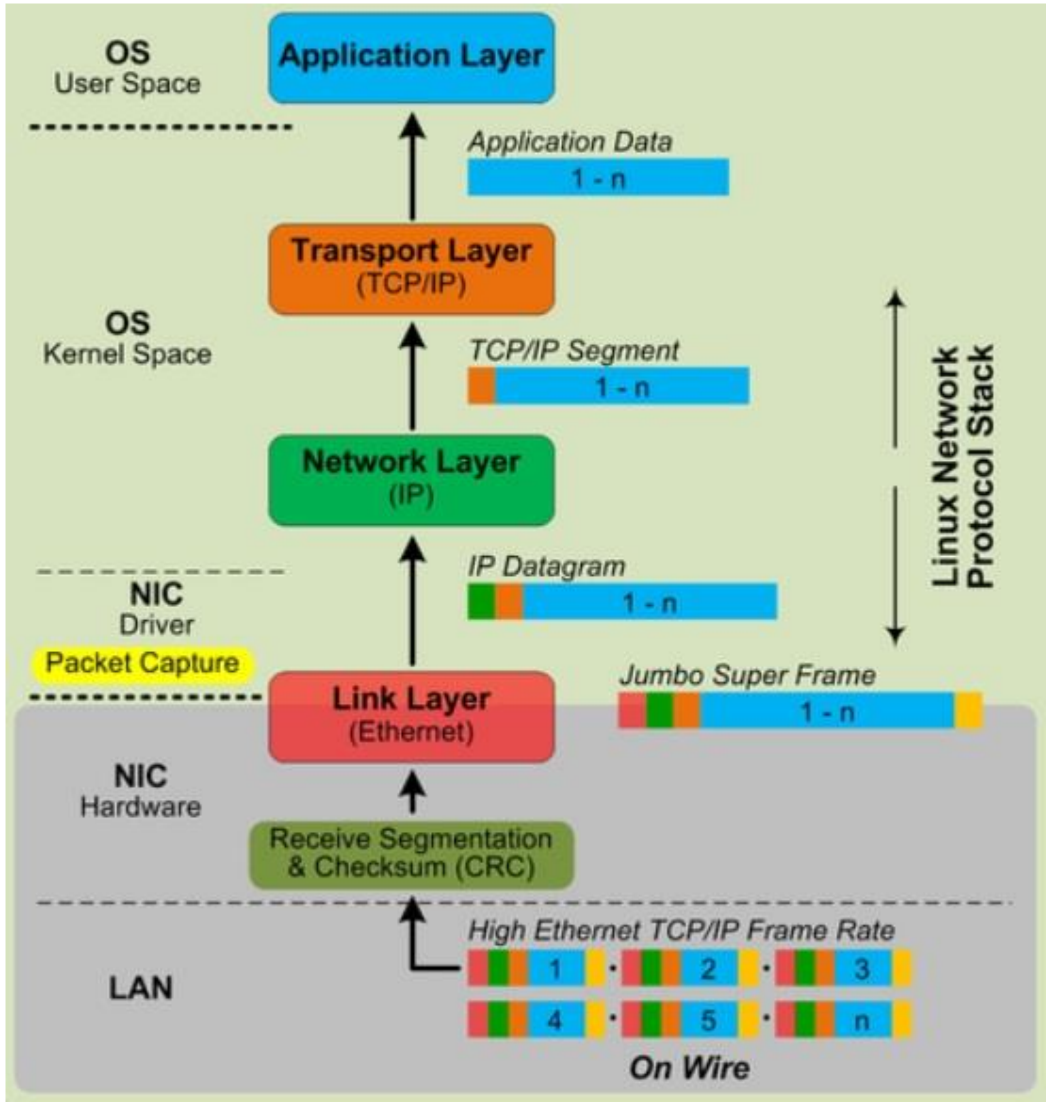
Receive Side Scaling (RSS)

- Receive side scaling (RSS) is a network driver technology that enables the efficient distribution of network receive processing across multiple CPUs in multiprocessor systems.
- A NIC uses a hashing function to compute a hash value over a pre-defined header fields of the received packet.
- A number of least significant bits (LSBs) of the hash value are used to index an indirection table.
- RSS improves the overall system performance by reducing:
    - Processing delays by distributing receive processing from a NIC across multiple CPUs.
    - Cache overheads since an incoming flow will be processed by the same core.



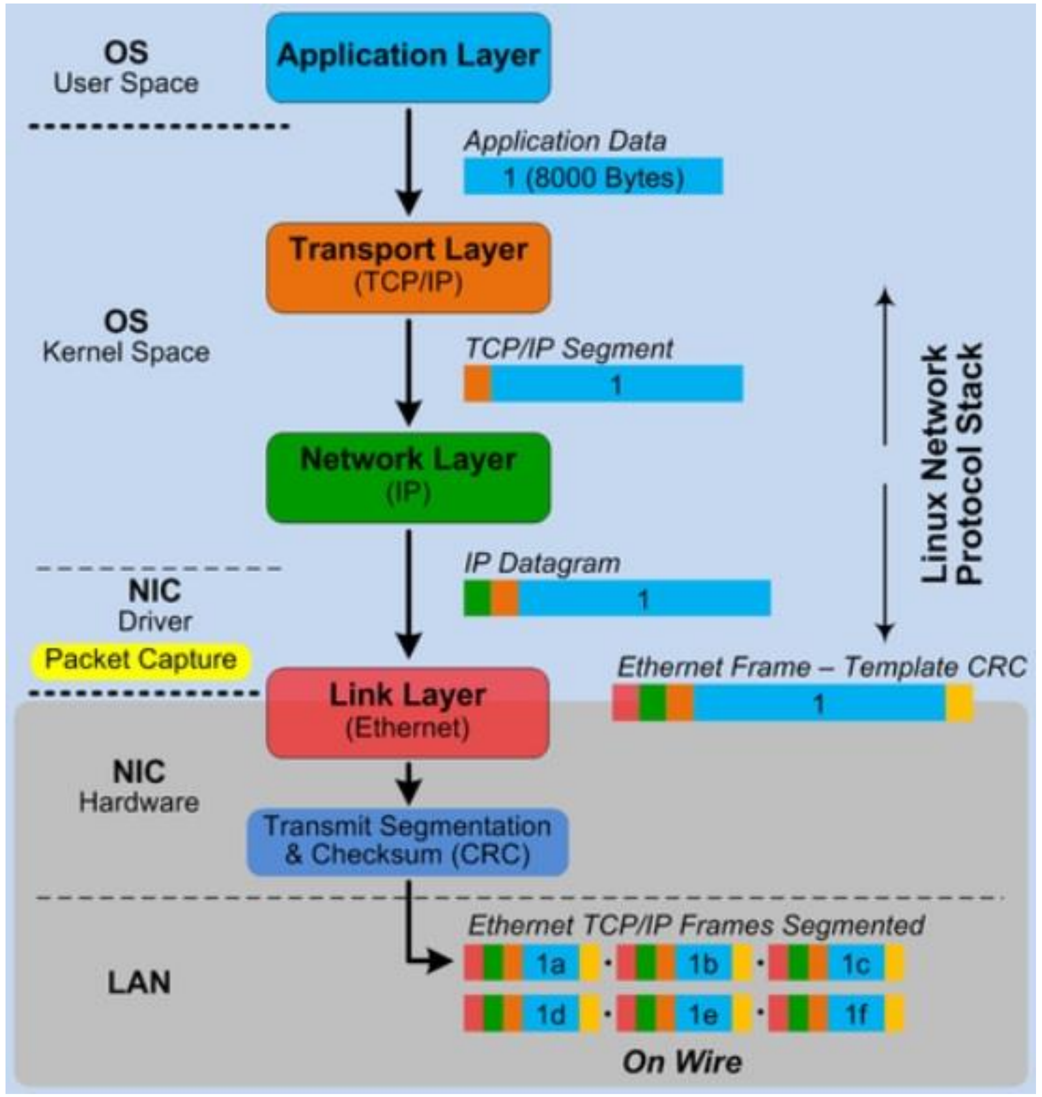
LRO (Large Receive Offload)

- Large Receive Offload (LRO) reduces the CPU overhead for processing packets that arrive from the network at a high rate.
- LRO reassembles incoming network packets into larger buffers and transfers the resulting larger but fewer packets to the host network stack.
- The CPU has to process fewer packets than when LRO is disabled, which reduces its utilization for networking especially in the case of connections that have high bandwidth.
- LRO is used for TCP.



TSO (TCP Segmentation Offload)

- Linux TCP stack sends a large TCP packet to the NIC. (much bigger than MSS).
- NIC performs the segmentation of this data based on the MSS value, adds TCP, IP and Ethernet headers to each segmented packet and sends the data on the wire.
- It saves host CPU cycles (don't need to segment TCP packets).



GSO and GRO

- Both GSO (Generic Segmentation Offload) and GRO (Generic Receive Offload) are performed on the host (not by the NIC!)

- GSO (Generic Segmentation Offload):
  - Uses TCP or UDP to send large packets
  - This is achieved by delaying the segmentation until as late as possible (for example when the packet is processed by the device driver).
- GRO (Generic Receive Offload):
  - Uses either TCP or UDP.
  - GRO is stricter than the LRO when reassembling packets.
  - It checks the MAC headers of each packet, which must match.
  - Only a limited number of TCP or IP headers can be different.
  - TCP timestamps must match.

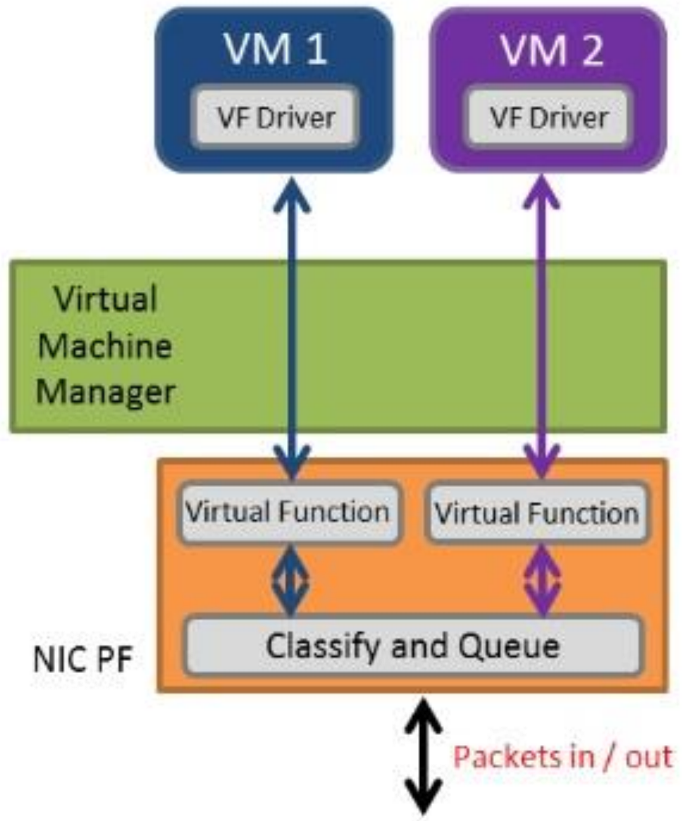
Accessing NIC from a VM

- Virtual Machine runs a guest OS with networking support.
- Guest OS can access networking functionality through two mechanisms:

1. SR-IOV:
   - Direct access to the physical device through VFs
   - No VMM/hypervisor involvement for I/O operations
2. VirtIO-net:
   - NIC device emulation through QEMU
   - Data copy is required (Guest user space to host kernel space and vice versa)

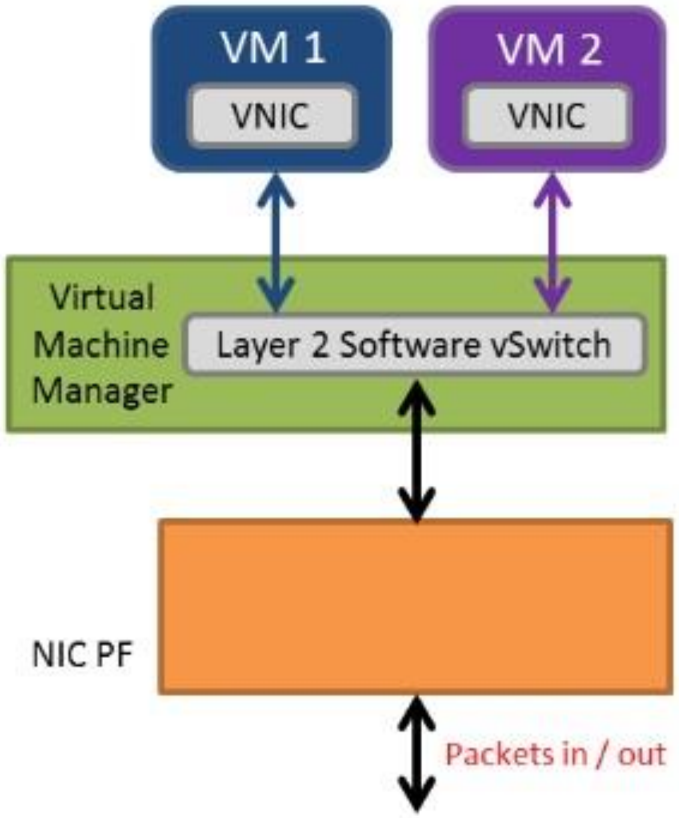
NIC – SR-IOV

- NIC VFs are exposed to the Guest OS – similar to physical devices
- Device VF driver is needed for the Guest OS
- VMM/Hypervisor is bypassed for all the I/O operations
- Device can DMA in/out from the Guest OS memory
- IOMMU is used for address translation
- Provides highest throughput, but requires VF driver to be present in Guest OS

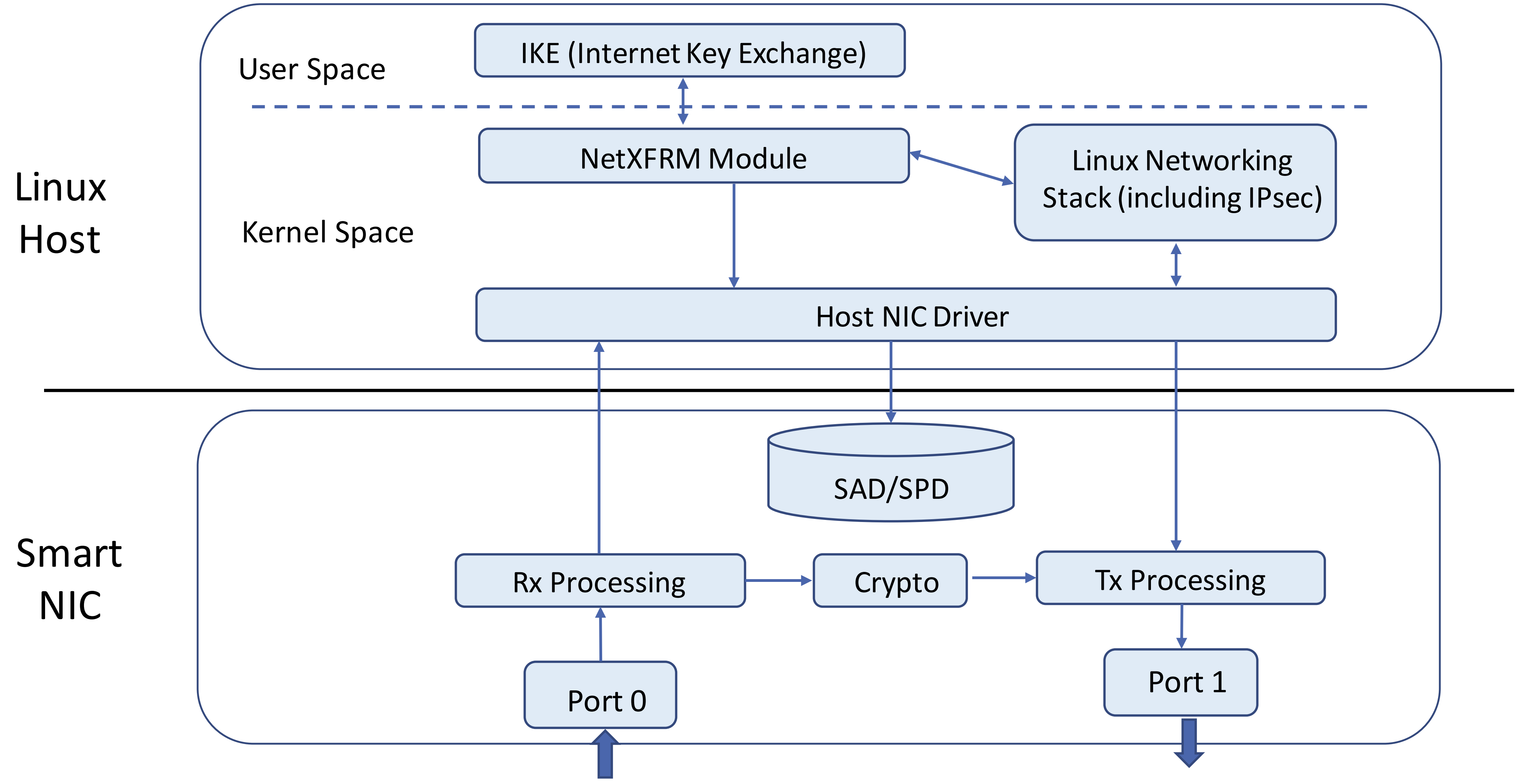


NIC – VirtIO-net

- Provides an emulated NIC device to a VM through QEMU.
- NIC driver has two components:
  - Frontend driver: virtio-net, resides in Guest OS.
  - Backend driver: vhost-net, resides in host kernel space.
- QEMU emulates a well known NIC device for which Guest OS has the NIC driver. No need for a physical NIC specific driver.
- Data copy is involved for I/O operation (Guest OS user space to host kernel space).
- More details on Virtio-net:
  - https://www.redhat.com/en/virtio-networking-series



NIC IPSec Offload



NIC IPSec Offload

- IKE protocol provides a mechanism to exchange and manage IPsec security associations which are used for IPsec.
- IKE daemon interacts with Linux kernel IPsec stack through PF_KEY sockets.
- Using PF_KEY interface, Linux IPsec stack can request IKE to negotiate a new SA.
- Once a new SA has been established through IKE, the SA data is pushed to Linux IPsec stack through the same PF_KEY interface.
- PF_KEY messaging infrastructure is defined in RFC 2367.
- We can create a custom PF_KEY kernel module (NetXFRM) to intercept messages sent by IKE to configure IPSec SAs and Policies.
- By intercepting these messages, SAs and Policies can be sent to the SmartNIC through the Host NIC driver.
- These IPsec SAs and Policies are also pushed to Linux IPsec stack so that both SmartNIC and Linux IPsec stack SPD/SAD remains in sync.