Authors: Authors: Subhashini Venkataramanan, Srini Addepalli 3/2015

Acceleration Interface for VNFs – IPSec and Packet Processor Use Case

Revision History

Date	**Version**	**Author**	**Reason**
03/05/2015	1	Freescale Semiconductor	Initial version

Authors: Authors: Subhashini Venkataramanan, Srini Addepalli 3/2015

1 Need Description

With NFVI (Network Functions Virtualization Infrastructure), Virtual Network Functions (VNFs) run as software-only entities in a hardware agnostic fashion. Examples of VNF range from

- Switching, Routing
- CDNs
- Security application such as Firewall, Intrusion Prevention systems, Virus and SPAM Protection Systems, IPsec and SSL-VPN gateways.
- eNodeB
- EPC SGW, PGW

While a range of VNFs may work efficiently as software-only entities, several of the VNFs such as IPS (Intrusion Detection and Prevention Systems), WAF (Web Application Firewalls that do virus scanning and spam protection), IPsec/SSL-VPN Gateways, LTE requiring Packet Data Convergence Protocol (PDCP) processing and VoIP (Voice over IP) Gateways do compute intensive algorithmic operations that takes away cycles off the VNFs. Achieving high performance for the above mentioned collective umbrella of Compute Intensive applications (CI) is a known challenge when run as VNFs.

Different CI VNFs require specific type of offload accelerators. The table below cites some examples of CI VNFs and the accelerators that they will need.

	VNF Application	Offload Accelerator Capabilities
1	IPsec/SSL Gateway	Symmetric Key Cryptography, Public Key Cryptography IPsec Protocol Accelerators, SSL Record Layer Accelerators
2	Intrusion Prevention Systems	Pattern matching, compression, decompression
3	Web Application Firewall, Anti-Virus, Anti-Spam Systems	Compression, decompression, pattern matching, SSL Record Layer Processing, Public and Symmetric Cryptography.
4	Packet Data Convergence Protocol	Crypto engines Protocol Acceleration
5	VOIP Gateway	Crypto engines SRTP Protocol Acceleration
6	Routing, Firewall	Table lookup Accelerators

The CI applications that run on propriety complex hardware-based physical appliances showcase higher performance as the compute intensive algorithmic operations (e.g. cryptography, compression/decompression, pattern matching) are offloaded to the hardware accelerators of SoCs. The major stumbling block in providing hardware acceleration for these CIs as VNFs is that the hardware accelerators available today have proprietary vendor specific interfaces that defeat the basic goal of NFV that envisages VNFs to be run as a software-only entity in a hardware agnostic fashion.

Keeping the requirement of VNF to achieve high performance virtualized network appliances which are portable between different hardware vendors, it becomes imperative to define a standard vendor independent accelerator interface, Virtual Accelerator Interface, so that VNFs shall continue to exist as software-only entities and work in a hardware agnostic fashion and yet address the performance challenges for the CI applications as VNFs.

In summary, the problem statement is as follows:

- CI VNFs are unable to showcase high performances as traditional CIs as they run as software-only entities. Using accelerators is one method with which CI VNFs can showcase higher performance as their traditional counter-parts.

- CI VNFs are unable to make use of hardware accelerators as they have proprietary vendor-specific interfaces and using such proprietary interfaces defeats the portability and migration requirements of VNFs across various ecosystems.

2 Proposal – Virtio based accelerators and Packet processors

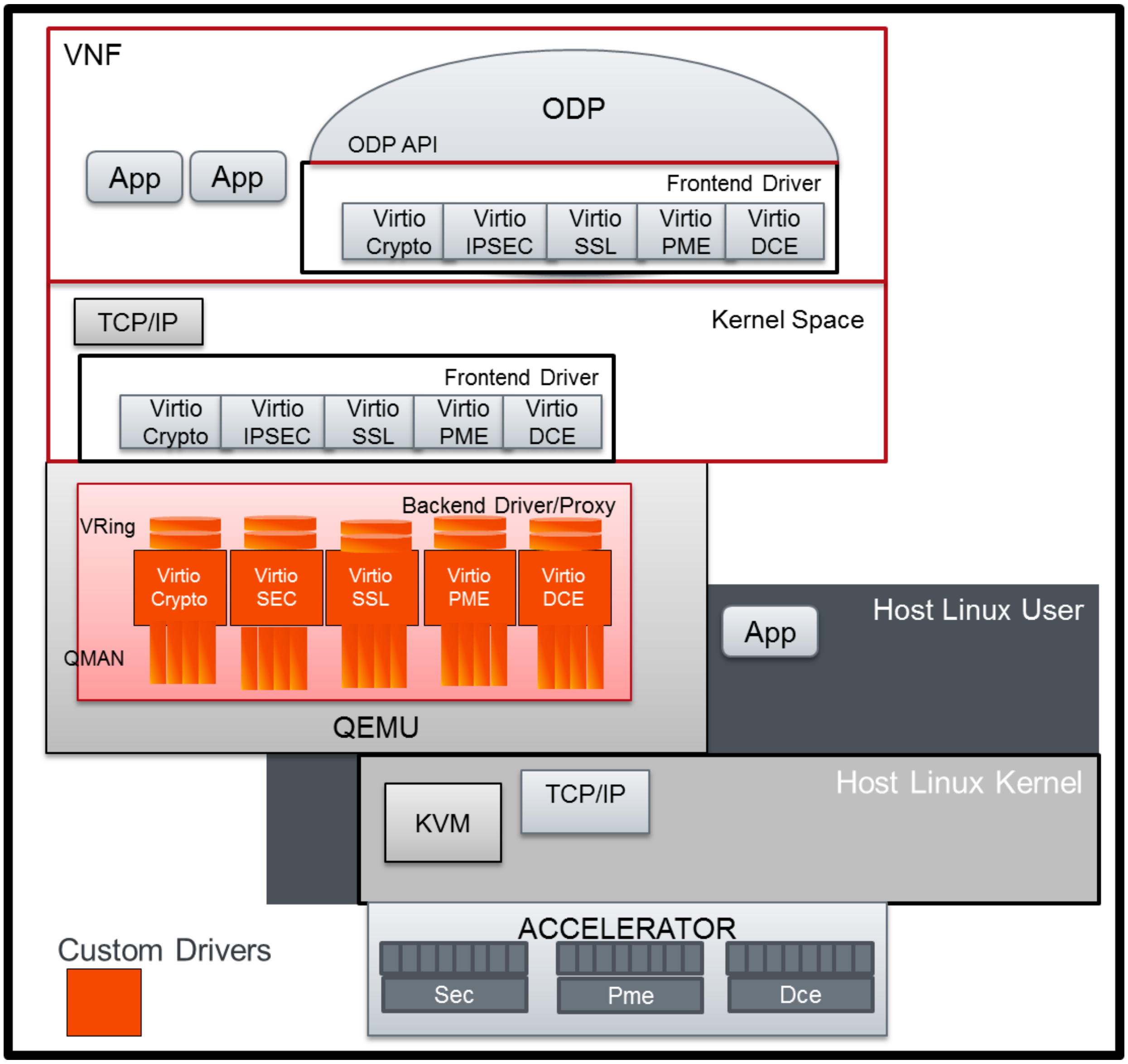

Figure 1: Standardized Accelerator Interface using virt-io drivers

Figure 2 shows a suggested implementation of standardized accelerator interfaces available to VNFs using Virt-IO drivers. VNFs can make use of the Virtio-Accelerator specific frontend drivers at the ODP level and the kernel level to access the underlying hardware accelerator.

Several classes of Virtio-accelerator drivers are shown in the picture – namely Virtio Crypto (for Crypto operations), Virtio IPsec (for IPsec level acceleration), Virtio SSL (for SSL level operations), and Virtio-PME for pattern matching acceleration and Virtio-DCE for compression, de-compression operations.

The backend of the Virtio drivers would include generic virtio class specific device with several back ends – one for each vendor that would have the capability to communicate with the underlying hardware accelerator. The purpose of the backend driver/proxy is to translate the messages in the Virtio-Virtqueue (Vring) descriptor to vendor hardware queue messages or descriptors, so that the underlying hardware accelerator can act upon the messages. When the hardware accelerator has finished processing the messages, the backend will make the responses available through the Virtio Descriptor Vrings to the frontend driver, which in turn announces to the VNF application.

2.1 Packet flow for IPsec Packet Processing

2.1.1 IPsec Packet Processing – Look Aside Accelerator Packet Flow

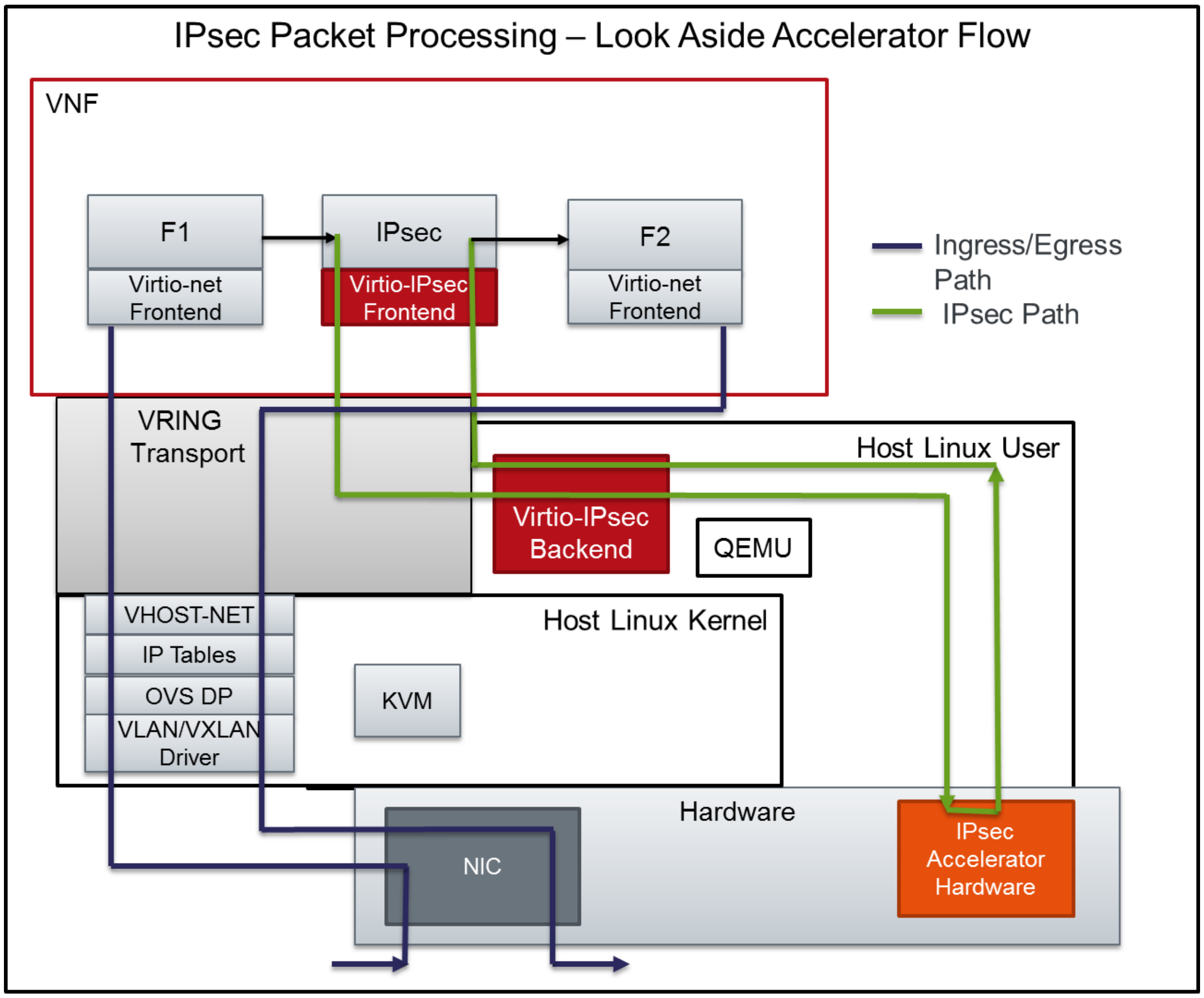

Figure 2 IPsec Packet Processing –Look Aside Accelerator Flow

Figure 3 shows the flow of packets when IPsec Look aside accelerator is used.

Ingress Packet Flow:

- Packets processed by VXLAN/VLAN, OVS Data Path, IP Tables, Vhost-Net
- Packet announced to VNF through Virtio-Net driver
- Packets under several function processing such as Firewall etc.
- Packets arrive at the IPsec module for IPsec Packet Processing

- As packets are submitted by the IPsec Module to the Virtio-IPsec front end driver, the buffers are put in the Virtio Descriptor Vrings or Virt Qs to be transferred to the Virtio-IPsec Backend.
- The Virtio IPSec Backend is responsible for translating the packets from Virt Q Descriptor to the actual hardware accelerator in a message that the accelerator understands and vice-versa.
- The Virtio IPsec Backend is also responsible for picking up processed packets from the hardware accelerator, updating the VirtQ rings and notifying the Guest VNF.
- The processed packets under further processing functions before being sent out through the Virtio interface.

2.1.2 NFVI Packet Processing Accelerator

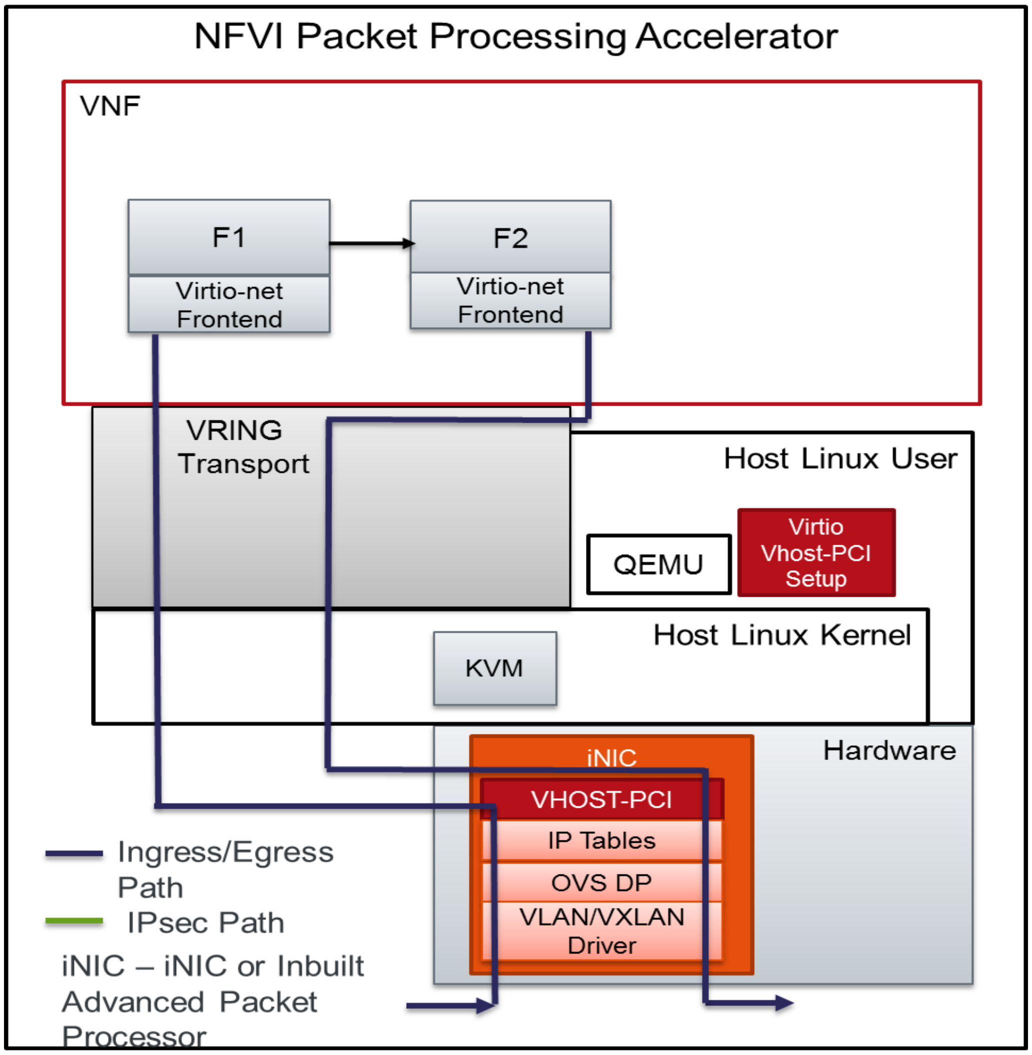

Figure 3 NFVI Packet Processing Accelerator

Figure 4 shows the packet flow for a NFVI Packet Processing Accelerator. The packet flow in this case is as follows:

- VXLAN/VLAN, OVS Data Plane, IP Tables processing happens in the Intelligent NIC (iNIC). The Vhost-PCI backend presents the packets to the VNF using the Virtio-net interface
- Packets that need to be transmitted out are submitted through the Virtio-Interface. The Vhost-PCI backend handles the packet, does the necessary processing before sending the packet out.

2.1.3 Combined NFVI Packet Processing with IPsec Look Aside Accelerator

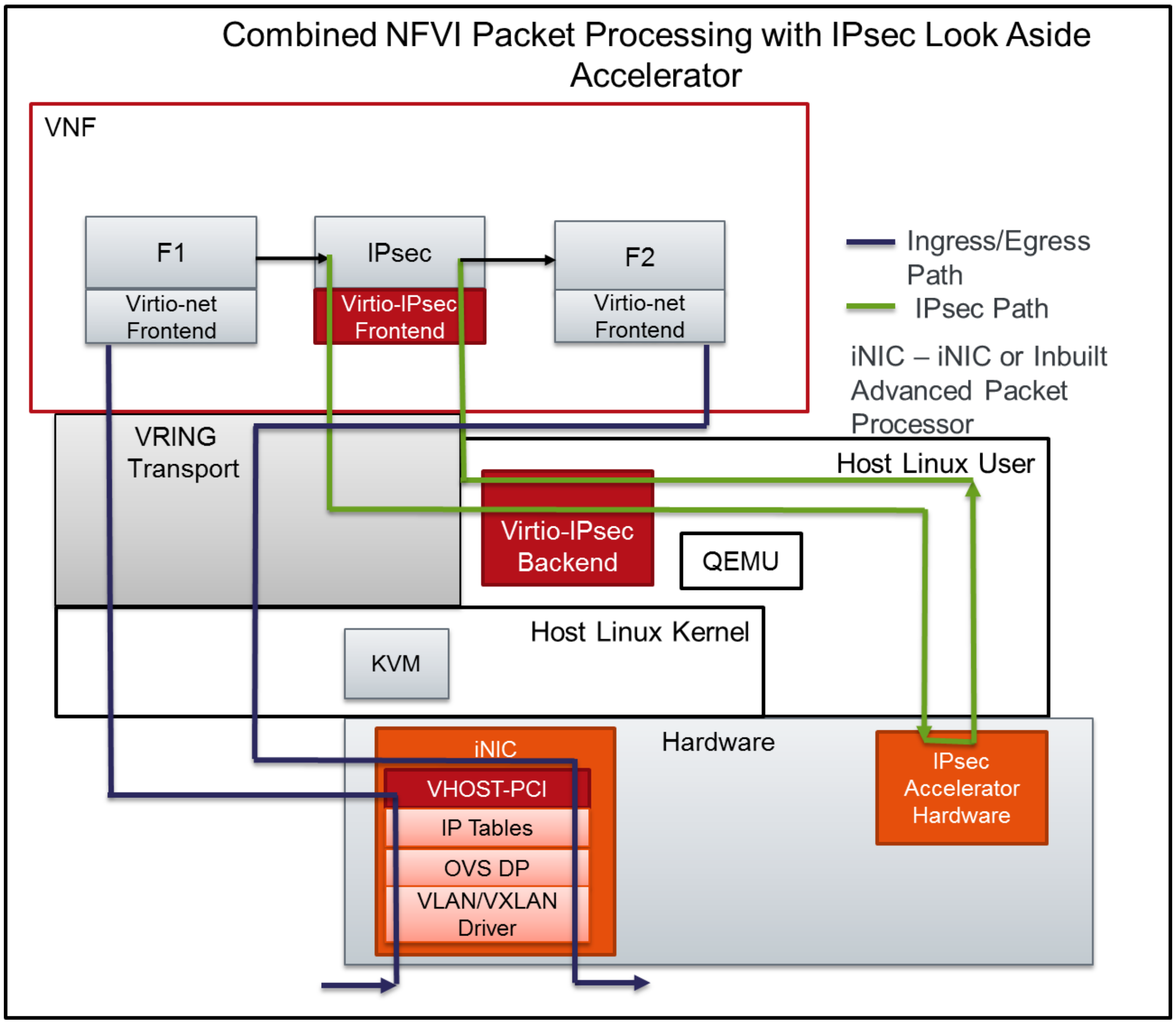

Figure 4 Combined NFVI Packet Processing with IPsec Look Aside Accelerator

Figure 5 shows the packet flow of a combined case of NFVI Acceleration and IPSec Look Aside Acceleration.

2.1.4 NFVI and IPSec Packet Processing Accelerator

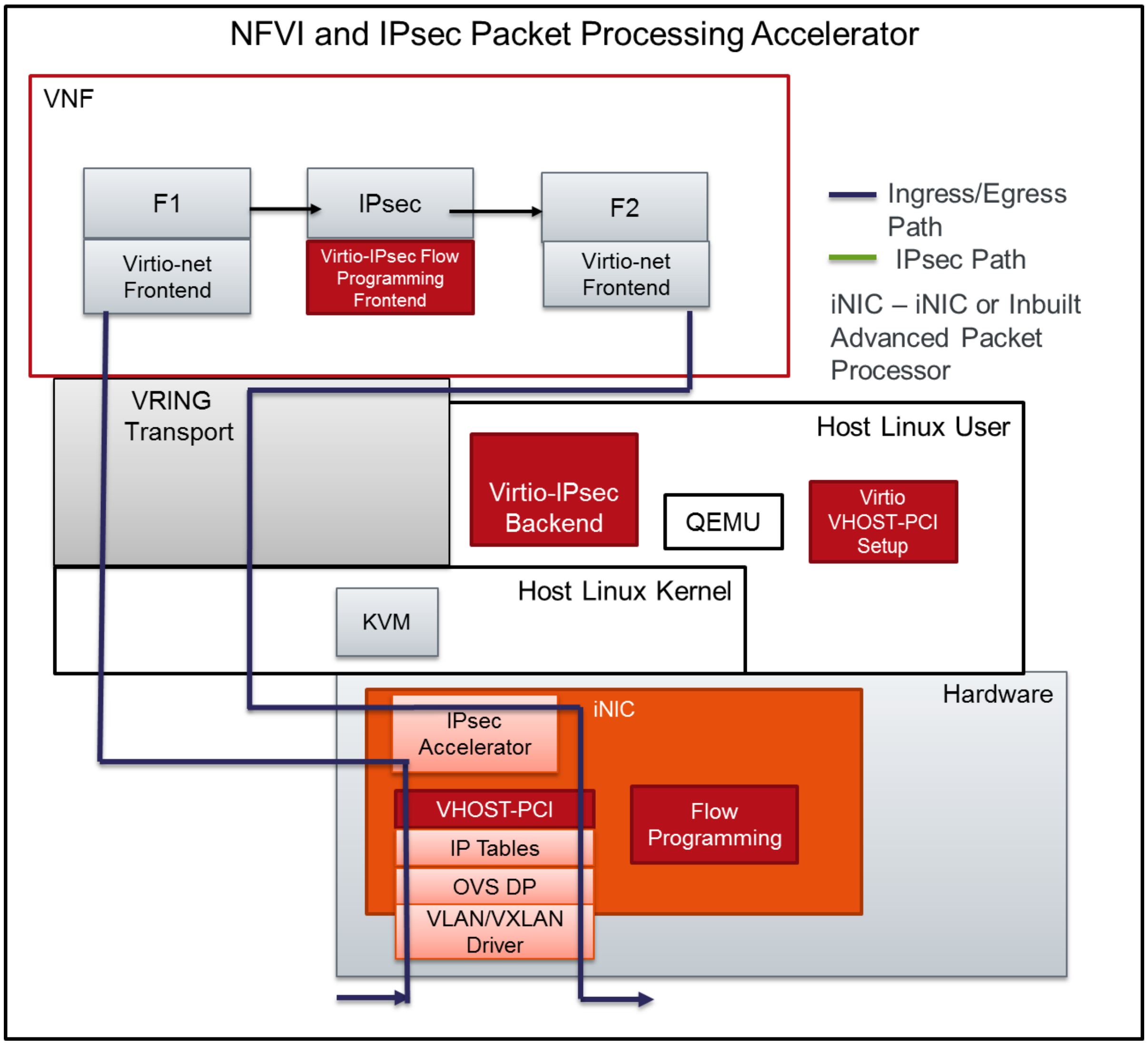

Figure 5 NFVI IPsec Packet Processing Accelerator

Figure 6 shows the packet flow, where the VNF is able to set up flow programming in the iNIC to apply IPsec Processing on specific flows. As a result of which, packets matching the programmed flows, destined to VNF are IPsec Processed (decrypted) before reaching the VNF. Similarly packets that match the programmed flows are IPsec Processed (encrypted) before being sent out by the iNIC.

3 Performance Benefits

- NFV Accelerator Offload using hardware accelerators can reduce significant compute cycles utilization by the VNFs leaving more for other tasks of VNF and hence result in a capacity gain, throughput gain, connection rate gain or some combination of the above for the VNF.

- Avoiding Virtualization Layer interaction on a per packet basis would bring significant performance gain and hence it is important to consider methods to bypass the Virtualization layer on a per packet basis.

4 Management & Orchestration Requirements

Orchestrator shall match VNF's accelerator's requirements with NFVI's accelerator capabilities. Some of the requirements are:

- Compute nodes shall advertise the accelerators it has, number of virtual entities accelerator can support, performance of the accelerators.
- Compute nodes shall periodically advertise the virtual accelerators and current bandwidth of accelerators.
- VNF images indicate the type of accelerators it can take advantage of.
- Orchestrator choosing the right compute node while instantiating the vNF.
- Orchestrator informing the compute node that is chosen to bring up the vNF with information on which accelerators are to be instantiated.
- Compute node to instantiate the accelerators and attaching them to the vNF that is being brought up.

5 Possible Accelerators

Crypto – Public key and Symmetric Key, IPsec Protocol Accelerator, SSL Record Layer Accelerators, Pattern Matching, Compression, De-compression, PDCP Accelerator, SRTP Protocol Accelerator and Table Lookup Accelerators

6 Live migration Consideration

- When VNFs move from one NFVI node to another, the VNF function should continue to work with minimal disruption.