Acceleration Interface for VNFs – IPsec and Packet Processor Use Case

**Revision History**

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

Table of Contents

[Table of Contents 3](#_Toc414437216)

[**1** Need Description 4](#_Toc414437217)

[**2** References 5](#_Toc414437218)

[**3** Classification 5](#_Toc414437219)

[3.1 Acceleration Models 5](#_Toc414437220)

[3.1.1 Look Aside Model 5](#_Toc414437221)

[3.1.2 Offload Model 5](#_Toc414437222)

[**4** Proposal – Virtio based Look aside and Offload Accelerators 6](#_Toc414437223)

[4.1 Look Aside Model 7](#_Toc414437224)

[4.2 Offload Model 8](#_Toc414437225)

[4.3 Packet flow for IPsec Packet Processing 10](#_Toc414437226)

[4.3.1 IPsec Packet Processing – Look Aside Accelerator Packet Flow 10](#_Toc414437227)

[4.3.2 NFVI Packet Processing Accelerator 11](#_Toc414437228)

[4.3.3 IPsec Packet Processing - Combined NFVI Packet Processing with IPsec Look Aside Accelerator 12](#_Toc414437229)

[4.3.4 NFVI and IPsec Offload Accelerator 13](#_Toc414437230)

[**5** Performance Benefits 14](#_Toc414437231)

[**6** Management & Orchestration Requirements 15](#_Toc414437232)

[**7** Possible Accelerators 15](#_Toc414437233)

[**8** Live migration Consideration 15](#_Toc414437234)

# 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.

# References

Virtio Specifications <http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.pdf>

# Classification

## Acceleration Models

There are primarily 2 kinds of acceleration models, namely Look aside and Offload.

### Look Aside Model

#### Accelerator Functions

Under the Look aside model, there can be several accelerator functions such as Crypto, IPsec, SSL, PME and so on.

### Offload Model

#### Accelerator Functions

Under Offload mode, there can be several offload functions supported such as Firewall, SLB, NAT, IPsec etc.

# Proposal – Virtio based Look aside and Offload Accelerators

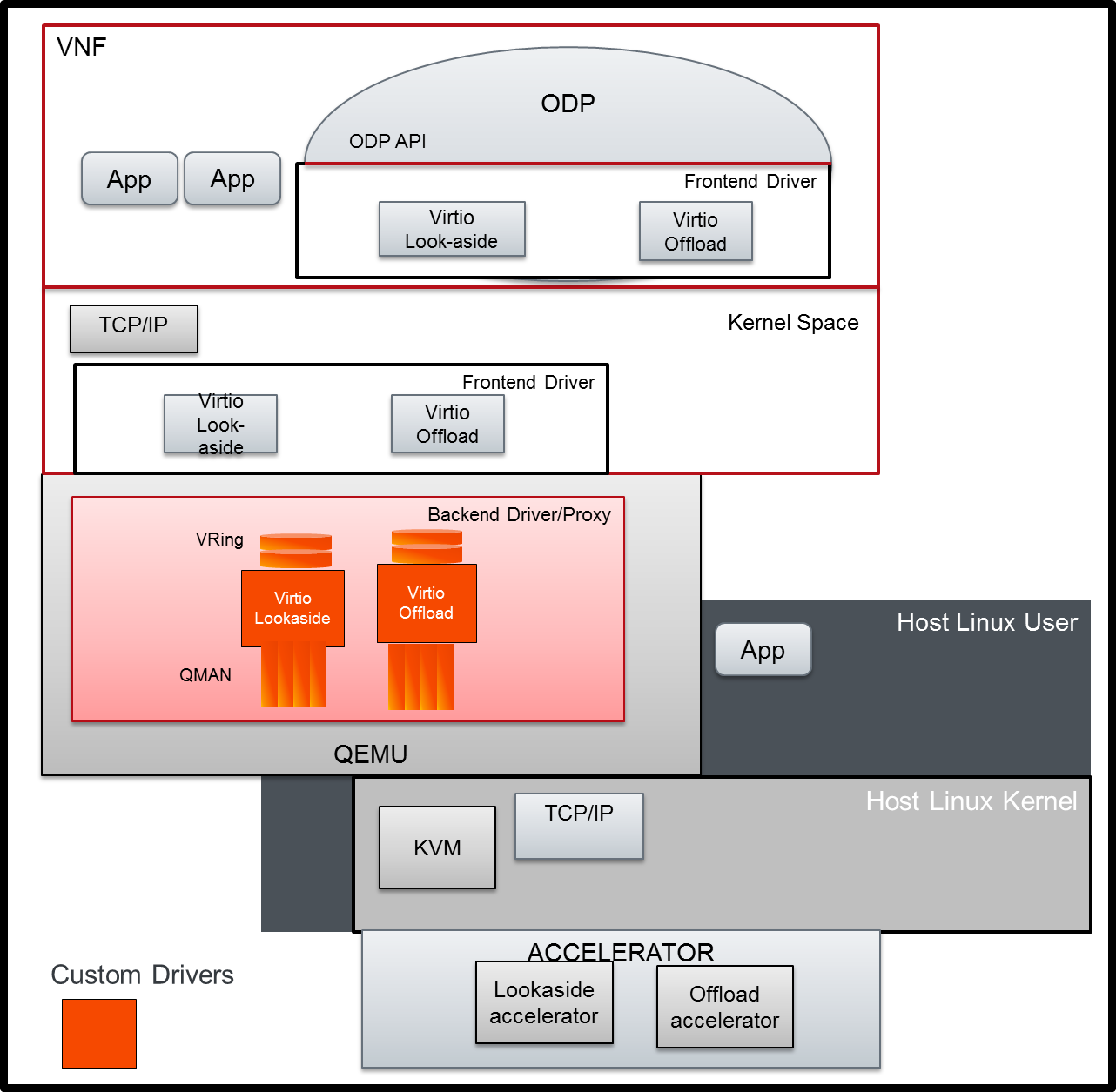


Figure 1 Virtio Interface for Accelerator

Figure 1 shows a suggested implementation of standardized accelerator available for VNFs using Virtio . Under the Virtio-lookaside model (umbrella of drivers), VNF can access several look aside accelerator functions such as IPsec, Crypto, PME, DCE etc. Under the Virtio-offload model (umbrella of drivers), the VNF can access several offload functions such as Firewall Offload function, SLB-NAT Offload function, IPSec offload function etc.

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.

## Look Aside Model

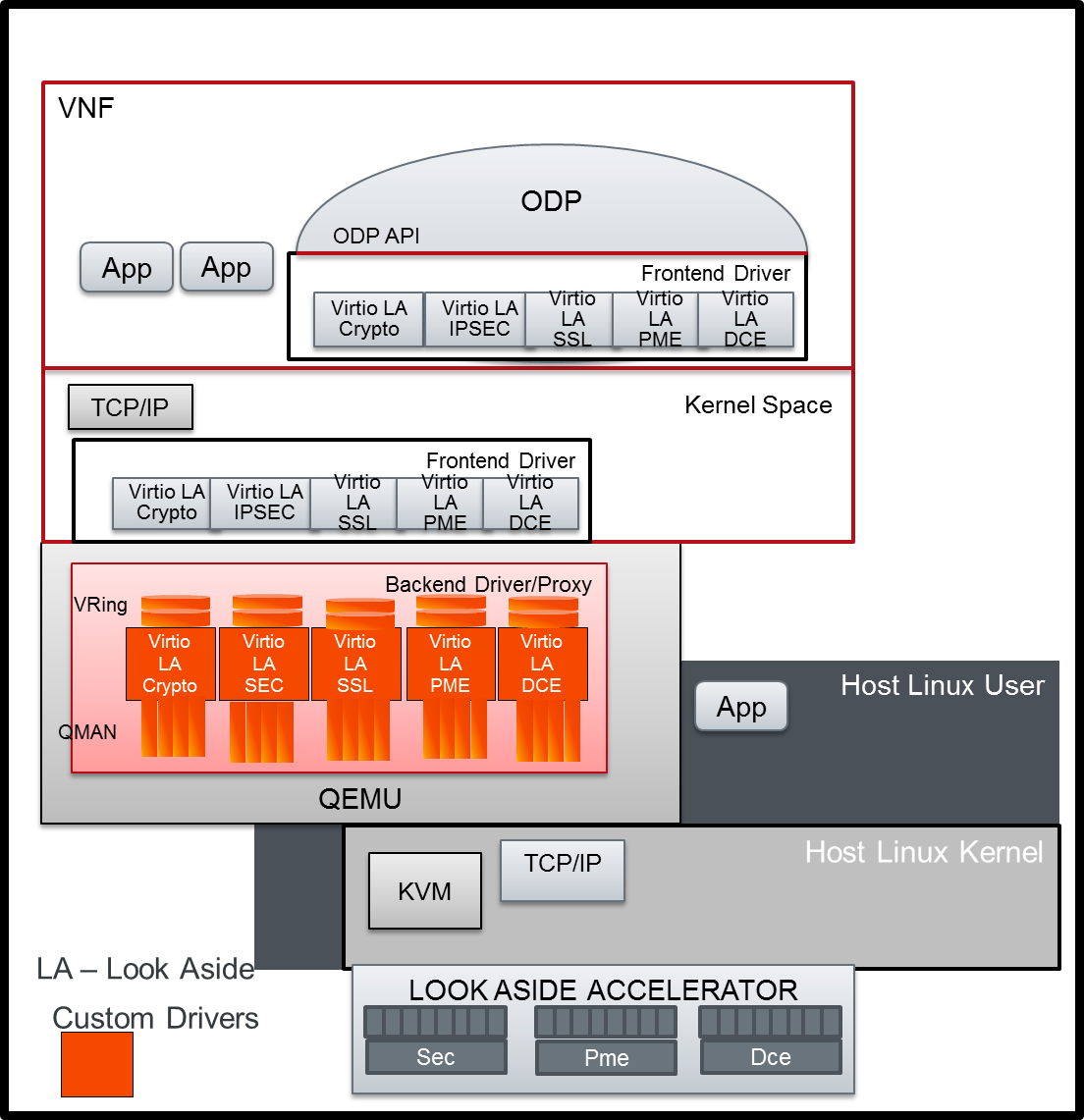


Figure 2: Standardized Look aside Accelerator Interface using virt-io drivers

Figure 2 shows a suggested implementation of standardized look aside accelerator interfaces available to VNFs using Virt-IO drivers.

Several Virtio-accelerator function drivers (LA or Look Aside model) are shown in the picture – namely Virtio LA Crypto (for Crypto operations), Virtio LA IPsec (for IPsec level acceleration), Virtio LA SSL (for SSL level operations), and Virtio LA PME for pattern matching acceleration and Virtio LA DCE for compression, de-compression operations.

The backend of the Virtio drivers would include generic virtio look aside function 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.

## Offload Model

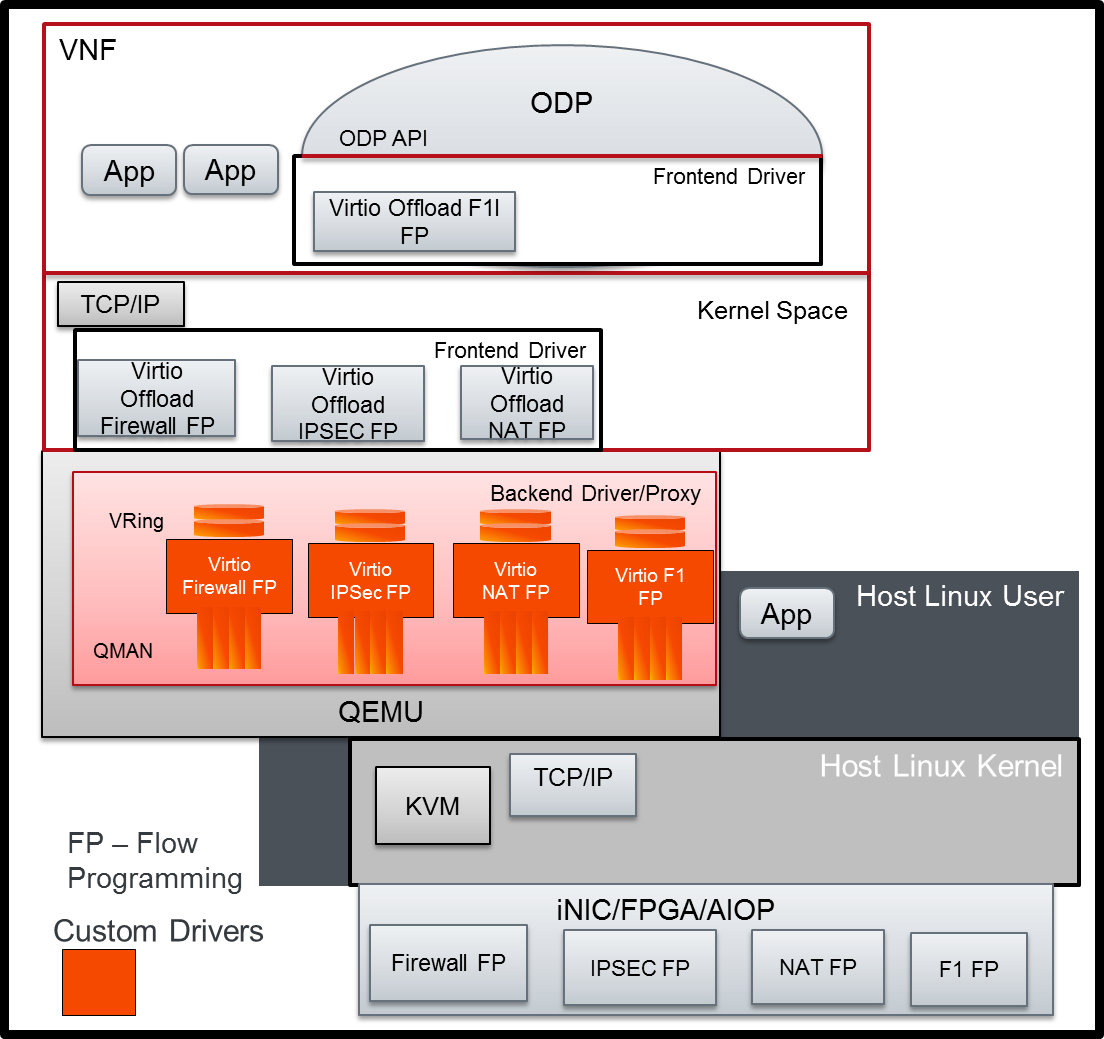


Figure 3 Virtual Accelerator Interface for Offload Model

Figure 3 shows a suggested implementation of standardized offload model interfaces available to VNFs using Virt-IO drivers.

Several virtio-offload function drivers are shown in the picture – namely Virtio Firewall flow programming, Virtio IPsec Flow Programming, Virtio NAT flow programming etc.

The backend of the Virtio drivers would include generic virtio offload function 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. Once the flows are programmed in the offload engines, the packet processing of those programmed flows will be handled by the hardware, without VNF intervention.

## Packet flow for IPsec Packet Processing

### IPsec Packet Processing – Look Aside Accelerator Packet Flow

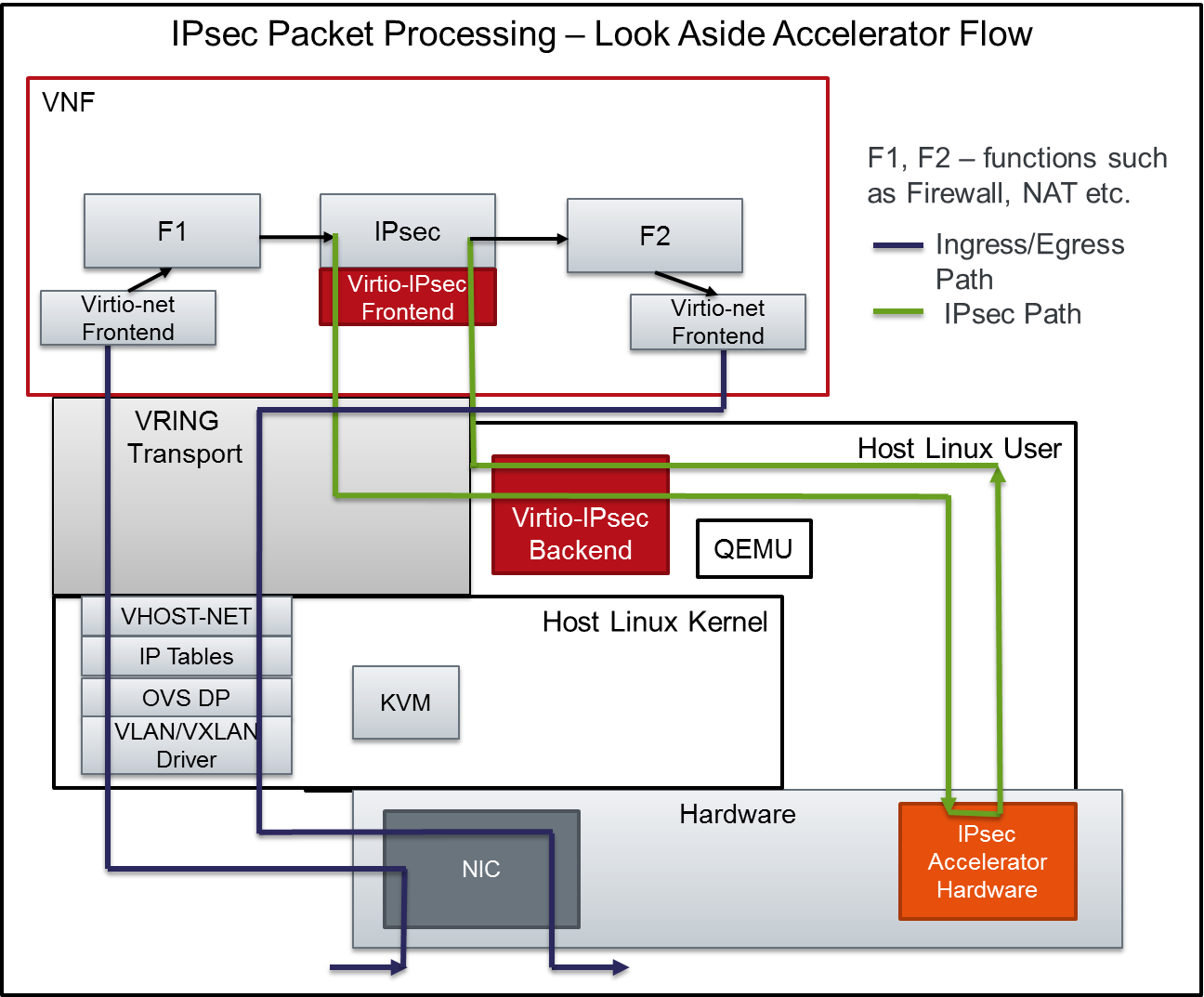


Figure 4 IPsec Packet Processing –Look Aside Accelerator Flow

Figure 4 shows the flow of packets when IPsec Look aside accelerator is used. F1, F2 stand for several packet processing functions such as Firewall, NAT etc.

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 (F2 etc.) before being sent out through the Virtio interface.

### NFVI Packet Processing Accelerator

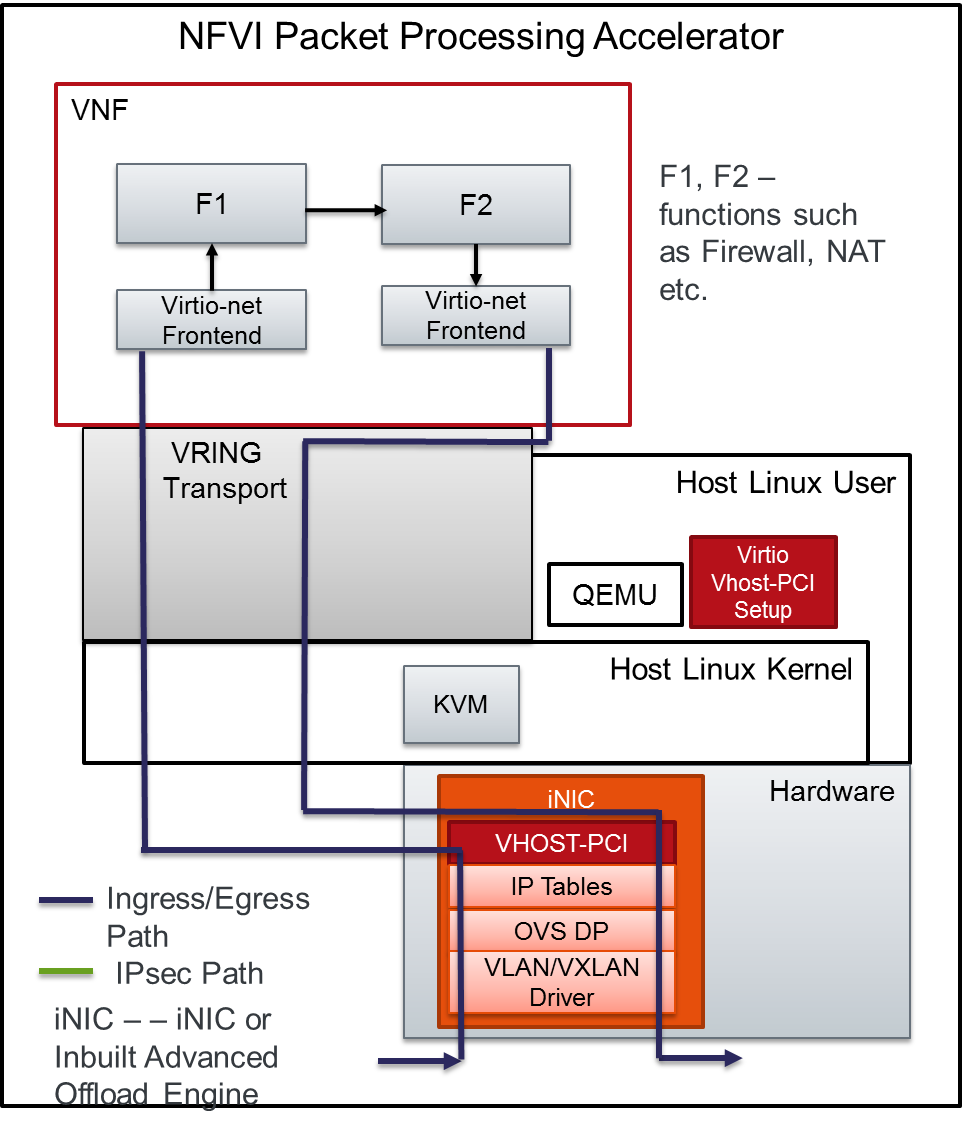


Figure 5 NFVI Packet Processing Accelerator

Figure 5 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.

### IPsec Packet Processing - Combined NFVI Packet Processing with IPsec Look Aside Accelerator

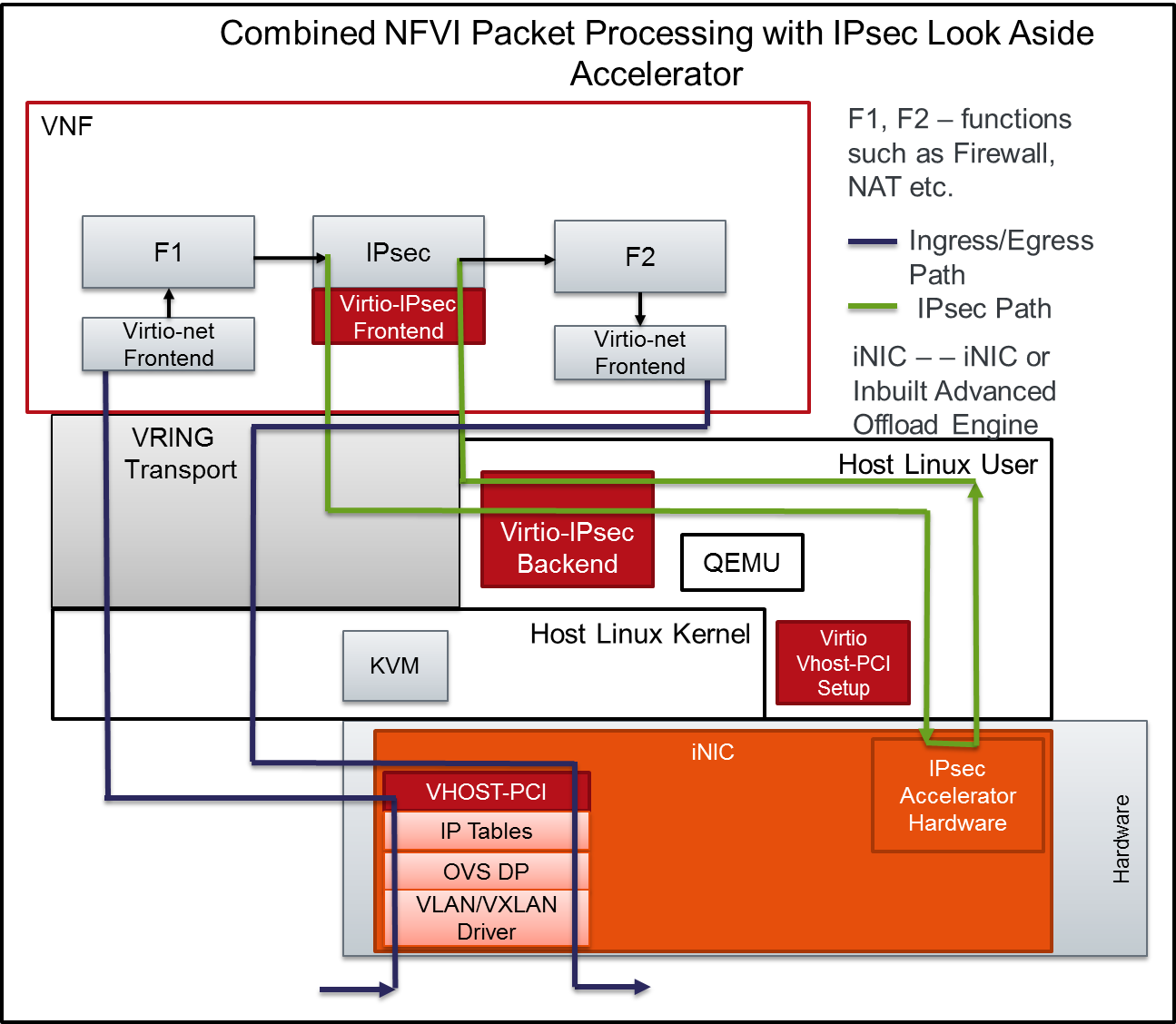


Figure 6 Combined NFVI Packet Processing with IPsec Look Aside Accelerator

Figure 6 shows the packet flow of a combined case of NFVI Acceleration and IPSec Look Aside Acceleration.

### NFVI and IPsec Offload Accelerator

#### Complete Offload – Packet flow

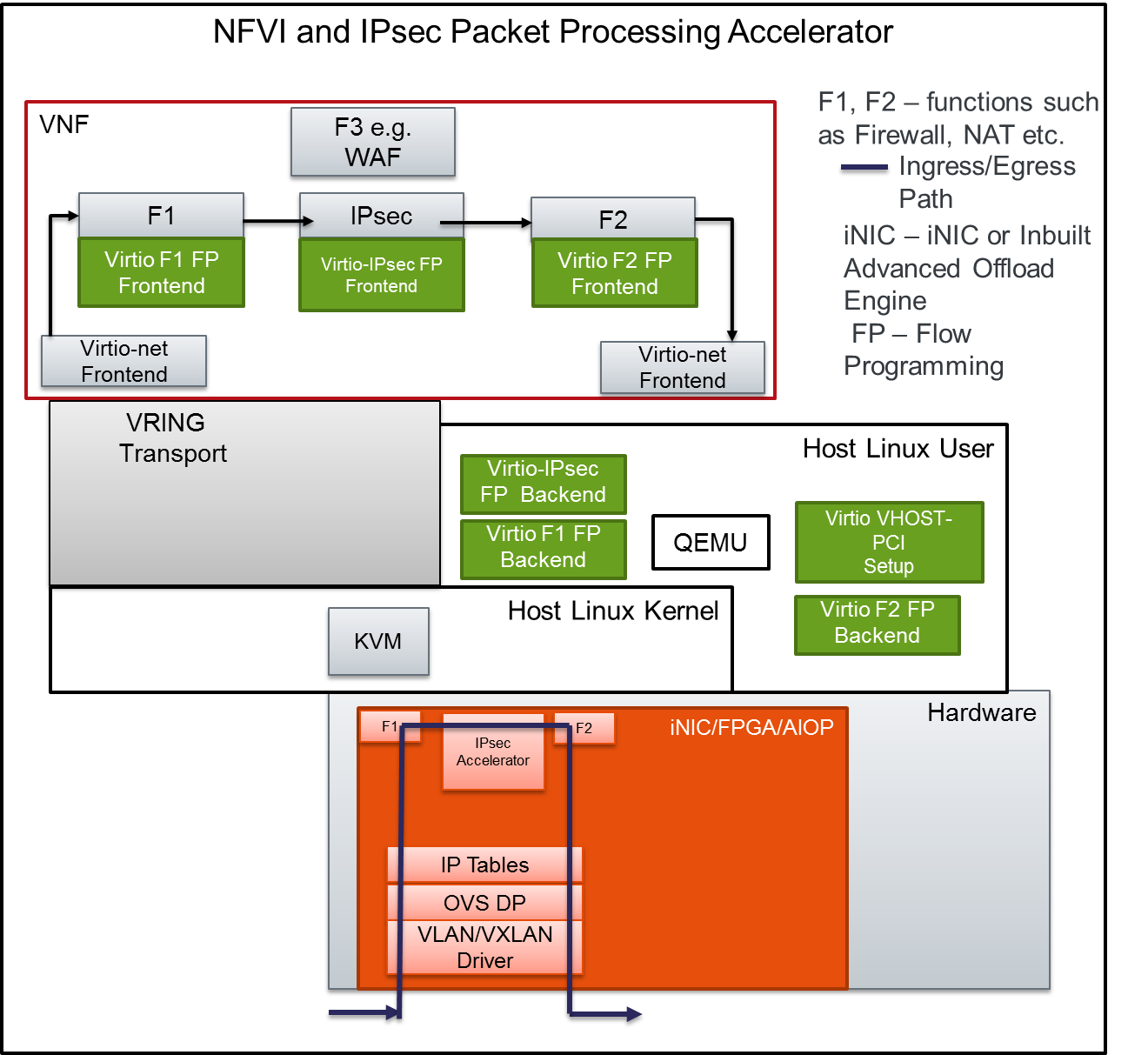


Figure 7 NFVI IPsec Packet Processing Accelerator – Offload Data Path Flow

Figure 7 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 this, for packets matching the programmed flows, packet processing happens in the iNIC itself.

#### Inline Offload – Packet flow

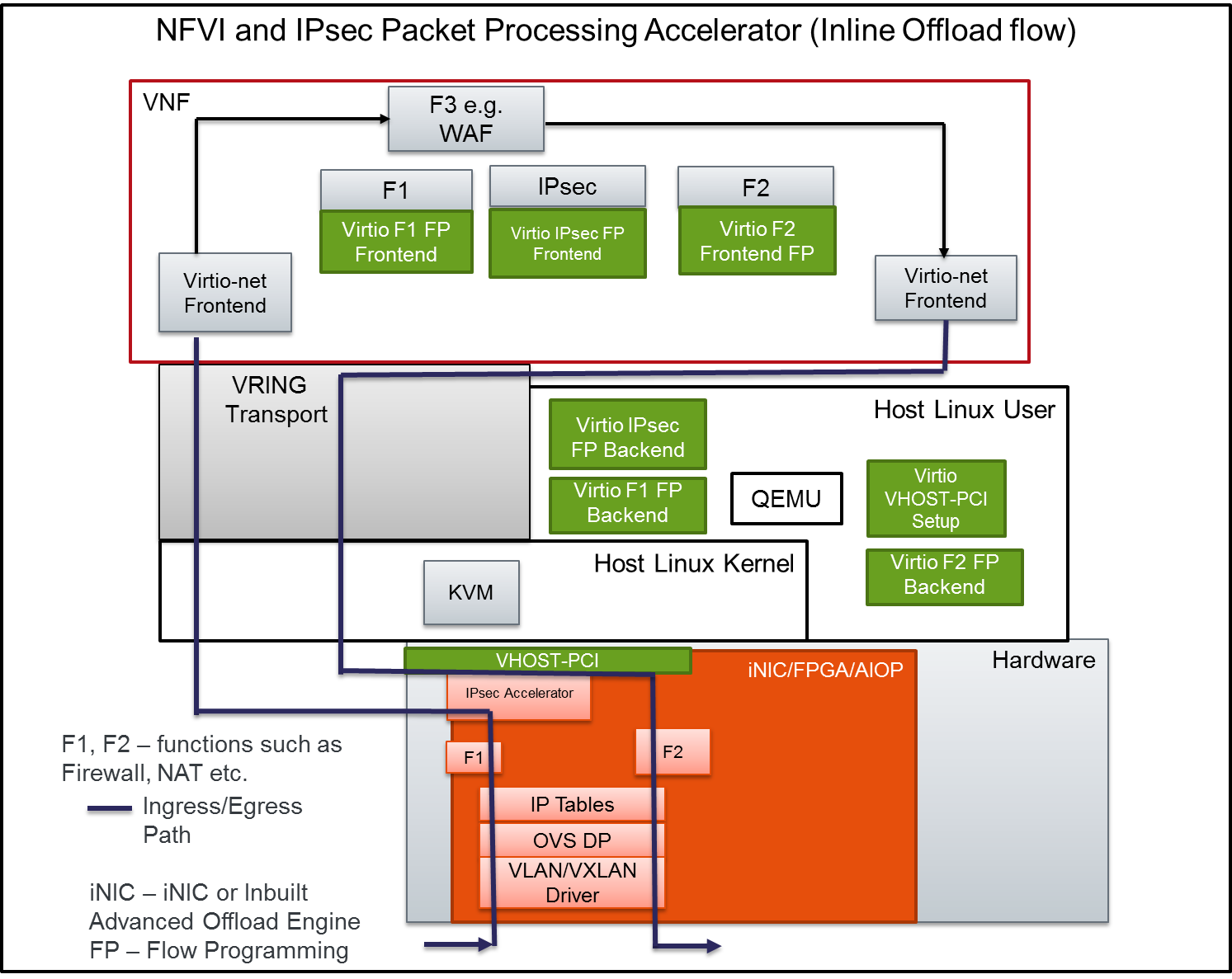


Figure 8 NFVI and IPsec Packet Processing Accelerator – WAF Flow (Inline offload)

Figure 8 shows the packet flow, in the case, where the packets have to be processed by the VNF e.g. TCP port 80 traffic. In this case, part of the packet processing happens in the iNIC as part of the offload operation, namely F1 processing and IPsec processing (Inbound processing or decryption). The packets are then sent to the VNF through the Virtio-net frontend to the F3 (WAF) for processing. Subsequent to processing by F3, the packets are sent via Virtio-net Frontend for subsequent processing (Outbound IPsec processing) and F2 before being sent out.

# 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.

# 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.

# 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

# Live migration Consideration

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