USE CASES FOR NFV APIs

# OVERVIEW OF NFV ACCELERATION API’s

Describe use cases for NFV API’s and also try and identify potential issues that need to be overcome to create API’s that work.

## Architectural Considerations

### Datapath considerations

In the NFV topology the VNFI will be running in the layer above the hypervisor and vSwitch. Hence the API’s defined need to take into account that all packets will be coming over a vSwitch that may have already done some data path offload. Hence there needs to be clear distinction drawn on what the scope of API’s need to be to accelerate in the VNFI vs what can be done in the vSwitch data path. It might be that both the vSwitch or the VNFI might be capable of acceleration of the same functions.

### Acceleration Implementation Considerations

Also there needs to be some method of provisioning acceleration and managing requests by Multiple VNFI’s to the underlying acceleration blocks such that the acceleration (HW or SW) can be shared amongst the VNFI’s with proper allocation enforced. Looking at other implementations such as vGPU or storage virtualization as a guide, this element should be included in the hypervisor, it should also cleanup any VNFI requests to the underlying HW blocks such that a single VNFI can be shutdown. This element may also need to save state to allow VNF migration.

Acceleration can be implemented as:

SW running on the same processor thread that issued the API request

SW running on a dedicated pool of threads other that the thread that issued the API request

Acceleration implemented as a fixed HW ASIC block on an SOC

Acceleration implemented as a fixed HW ASIC block on a bus attached (PCI, PCIe, QPI) accelerator with larger request/response latencies

Acceleration implemented as a reconfigurable block i.e FPGA, Software, GPU on the same SOC

Acceleration implemented as a reconfigurable block i.e FPGA, Software, GPU on a bus attached (PCI, PCIe, QPI) accelerator with larger request/response latencies

Acceleration can be inline to the packet datapath or can be a lookaside implementation

## API Design Considerations

We should determine factors effecting API design:

asynchronous vs synchronous

state full vs non-state full offloads

Usage of callback’s, interrupt driven or pollmode driven

# CRYPTOGRAPHY ACCLERATION USE CASES

## IPSEC Symmetric Crypto Offload

Offload of decryption or encryption of IPSec packets. This acceleration can be broken down into several individual actions:

### IPSEC Decrypt:

Uses cipher key, cipher algorithm and data to decrypt the IPSEC payload. Result is the cleartext IP packet that is sent up the networking stack for further packet processing.

### IPSEC Encrypt:

Use cipher key, cipher algorithm and data to encrypt the IPSEC payload. An IPSEC header is than added and the packet is transmited immediately.

### Datapath Considerations for API design

For crypto accelerators API’s can be implemented in a look-aside manner, this means the API sends the data, cipher type and key to the accelerator and then gets a result back, this works well when the accelerator is attached over a bus and is not inline with the packet data path and does not have packet processing capabilities.

The crypto accelerator API’s can be implemented also inline to the data path. In this mode the API sends the cipher type, key and flow identifier to the accelerator. The accelerator stores the cipher and key and received a stream of packets from the packet ingress and for every packet that matches the flow identifier the accelerator will perform the encryption or decryption, in the encryption case the accelerator will assemble the entire IPSEC packet and pass it to the egress path and the VNFI will not need to be involved further. For the decryption case the accelerator might have to take some responsibility for IP Packet fragment reassembly

## IPSEC/TLS/SSL PKI Offload

Accelerate public key exchange. Typically is done though look aside accelerator API’s

## TLS/SSL Symmetric Crypto Offload

Offload bulk crypto offload for decrypt or encrypt of the TCP payload.

### SSL/TLS Decrypt:

Uses cipher key, cipher algorithm and data to decrypt the TCP segment. Result is the cleartext TCP Segment that is sent back to the HTTP layer.

### SSL/TLS Encrypt:

Use cipher key, cipher algorithm and data to encrypt the TCP segment. The encrypted TCP segment is transmitted using the TCP protocol stack and the IP protocol stack..

### Datapath Considerations for API design

# REGULAR EXPRESSION ACCLERATION USE CASES

A regex engine can be used to implement state machines that match an incoming pattern against a table of patterns to see if the pattern matches any of the patterns in the table.

## Deep Packet Inspection (DPI)

DPI typically uses the first three hundred bytes of a TCP Segment to identify the application or protocol. DPI is typically used to identify which website a user is navigating to or which application (such as torrent, skype, whatsapp) is sending data over a network.

## Intrusion Protection and Detection (IPS/IDS)

IPS/IDS is used in firewall’s to detect potential security threats that can come as attachments and files over email or other network medium. The IPS/IDS engine might have to scan all the bytes of an email attachment to check for a potential virus or security threat that is residing in the attachment.

## Datapath Considerations for API design

API design for a regex engine can be implemented in a non-statefull manner. In this case the DPI or IPS/IDS engine will assemble the entire file that needs to be inspected and sends the entire data to the regex engine in one large chunk. When a new file needs to be inspected the regex engine can start scanning the file assuming that there is no correlation between the new file and the previous file that it received.

API design for a regex engine can be implemented in a statefull manner. In this case the DPI or IPS/IDS engine will send segments of a file as they are received to the IPS/IDS engine. When the regex engine receives a chunk of a file that it was previously working on, it must reload the state engine from where it was left off and pick up its scanning of the next chunk of data such that the chunks were just one large continuous file. In this case the Accelerator will need to store the state machine states between chunks of data.

# tCAM ACCELERATION USE CASES

tCAM is a special memory used to find policy lookup in a table much faster than can be implemented in a software implementation.

## Firewall Policy Lookup

When a new packet from a flow enters a firewall, the firewall mush make some decisions on the packet such as if it it should allow the packet to pass though, forward the packet, drop it, or do header modifications. A large policy table needs to be traversed to find the exact policy that needs to be implemented for the packet.

## Router LPM Lookup

When a new packet from a new flow enters the router tCAM is used in routers to implement policy look up to determine how to handle the flow, the policy typically includes header modification before the packet is transmited. A large policy table needs to be travered to find exactly how to handle the packet.

## Datapath Considerations for API design

# VIDEO TRANSCODING ACCELERATION USE CASES

# PACKET DATA ACCELERATION USE CASES

Accelerate packet datapath using common data path accelerators such as buffer managers and traffic managers.

vSwitch Acceleration

Packet data path accelerators can be used to accelerate the entire packet data path from the ingress Ethernet ports though the vSwitch to the VNFI and back out. A buffer manager can manage the buffers as they pass thought the vswitch and into the VNF and effectively offload the vSwitch and the VNFI from allocation of buffers and freeing buffers and otherwise managing a pool of buffers.