Data Plane Acceleration

Lingli Deng, China Mobile, fuqiao@chinamobile.com
Denis Crasta, Freescale

denglingli@chinamobile.com
denis.crasta@freescale.com
Outline

- Background: Smallcell GW VNF for CMCC
- Data Plane Acceleration Architecture and Requirement of NFV
  - Define a common architecture of leveraging data path acceleration capabilities for VNF and NFVi.
  - Specify and roles and requirements for each of the components identified and interfaces between them
  - Scenario and usecases and test spec
- Introduction to demos of proposed implementations
  - Onsite demos:
    - Intel booth: DPDK port to ARM SoCs
    - ODP Summit: ODP for X86 and ARM SoCs
  - Ongoing work:
    - virtio-ipsec-LA
    - IOVisor for vSwitch
SmallCell GW sits at the edge of EPC, on the path between smallcell and core network. It has two major components: a signaling GW (SmGW) and a security GW (SeGW). The SmGW is an optional component, while the SeGW is a mandatory component.

**SeGW**
- **Authentication**: realize mutual authentication between small cell and GW.
- **Security Protection**: establish IPSec tunnels between small cell and GW.
- **QoS Inheritance**: copies the inner IP ToS/DSCP tags onto the outerIP header during encapsulation.

**SmGW**
- **Signaling Routing**: selects a proper MME for an attaching UE.
- **Signaling Pooling**: pools the interfaces to MME for a large group of small cells.
- **Optional**
Background
NFV Smallcell GW

Service Layer
- Independent VNF software development and upgrades on top of fully-decoupled hardware platforms.

Platform Layer
- Simplified deployment, flexible scheduling, on-demand scaling due to hardware virtualization.

Hardware Layer
- Reduced Capex/Opex for off-the-shelf IT devices.

Virtual Machines
- SmGW VNF
  - SW acc
- SeGW VNF
  - HW acc driver
- other VNFs

Intel Reference COTS Server

NFVO
VNFM

VIM
Common interfaces for DataPlane Acceleration

- HW/SW acc is common in dataplane devices
  - Encryption, DPI, transcoding, firewall, etc.
- Concern: no common interfaces exist for acceleration
  - e.g. for IPsec acceleration
    - software-only acceleration
    - hardware assist accleration in various forms:
      separate acc card/integrated with NIC/integrated with CPU
  - What happens if the ISP chooses another acceleration
    - VNFs: rewrite their code extensively to do hardware migration
    - ISPs: undesirable binding between VNF software with the underlying hardware devices
- Vision: a fully decoupled architecture
1. Common interfaces on the data plane to enable VNF portability across accelerators/platform/ISAs (VNF/NFVi)
2. Enable accelerator discovery and monitoring on the local compute node or remote acc pool (VIM/NFVi)
3. Enable acc resource orchestration and monitoring (NFVO/VNFM)
4. Define the hardware acceleration requirements for SeGWs and SmGs

1. Add a specialized hardware chipset
2. Utilize a set of general APIs to get access to the hardware chipset
3. Allocate SeGWs to "capable" VMs on top of "capable" devices
4. Define the hardware acceleration requirements for SeGWs and SmGs
Target:

- specify detailed framework/API design/choice and document test cases for selected use-cases;
- provide open source implementation for both the framework and test tools;
- coordinate integrated testing and release testing results; and
- interface specification.

Current Work items for B-release

- framework: architecture and requirements
- usecases
- gap analysis
- test spec (collaborate with testing projects)

Related Upstream Projects

- DPDK, OpenDataPlane, OpenCL, libvirt, virtio
DPACC: Acceleration Layer for Guest (VM's)

**Virtual Machine or Guest**

**VNF Application**

- **g-API**: Generic API for application portability (required)
  - API as a standard application interface and is required
- **legacy-API**: Legacy API for applications (Optional)
  - The legacy-API layer is to help existing applications using APIs like sockets, libcrypto, ...
- **s-API**: APIs for utilizing an AC (APIs from the AC)
- **sio**: software I/O interface
  - e.g. VirtIO to the underlying SW/HW
  - Enables binary compatibility with paravirtualized drivers
  - Optional for hio-only requirements
- **g-drivers**: General driver for each device type
  - Implemented in software or the frontend to the hardware (may be different for different acceleration functions)
- **hio**: Hardware I/O interface
  - Some type of pass-through design with support for virtualization (e.g. SR-IOV, SoC-specific interfaces, etc.) Optional for hio-only deployments

**Software Acceleration Layer (SAL)**

- Provides a target abstraction for application software
- **g-API**: Generic API for application portability (required)
- **legacy-API**: Legacy API for applications (Optional)
- **s-API**: APIs for utilizing an AC (APIs from the AC)
- **sio**: software I/O interface
  - e.g. VirtIO to the underlying SW/HW
  - Enables binary compatibility with paravirtualized drivers
  - Optional for hio-only requirements
- **g-drivers**: General driver for each device type
  - Implemented in software or the frontend to the hardware (may be different for different acceleration functions)
- **hio**: Hardware I/O interface
  - Some type of pass-through design with support for virtualization (e.g. SR-IOV, SoC-specific interfaces, etc.) Optional for hio-only deployments

**If s-API is AC specific APIs and cannot provide portability across platforms, then the g-API is mandatory to ensure portability**
DPACC: Acceleration Layer for Host (Hypervisor)

SAL: Software Acceleration Layer
Provides an abstraction between SW and HW
- **sio-backend**: backend of paravirtualized drivers
  - **vHost-user**: User space based VirtIO interface
    - optional for VM ←→ host access
- **s-API**: APIs for utilizing an AC (APIs from the AC)
- **g-drivers**: General driver for each device type
  - Implemented in software or the frontend to the hardware (may be different for different acceleration functions)
- **hio**: Hardware I/O interface
  - Non-virtualized, accessed only by host SAL

AC: Software/Hardware Acceleration Core
- e.g., DPDK, ODP or other acceleration implementation

SRL: Software Routing Layer (Optional layer for the host)
- Open vSwitch (OVS) or vRouter

AML: Acceleration Management Layer
- To be define for orchestration and spans more than the SAL
Dpacc implementation PoCs

- **Onsite demos**
  - Intel booth: DPDK port to ARM SoCs
    - upstreamed to DPDK
    - initiated by Intel and followed by Cavium and Freescale
    - more input from ARM silicon vendors to be expected
  - ODP Summit: ODP for X86 and ARM SoCs
    - ODP Mini Summit
    - AMD, Freescale, EZchip demos

- **Ongoing work**
  - virtio-ipsec-LA
  - mgmt enhancements for Openstack NOVA

- **New proposal**
  - IOVisor for vSwitch
Join us in Dpacc

- Join us on the usecases and test spec if you are intend to deploy data plane VNFs on your network
- Join us on the interface requirements and PoC implementation if you are developing a VNF
- Join us on the mgmt requirements and PoC implementaiton if you are developing a NFVi or NFV MANO

Project Page: [https://wiki.opnfv.org/dpacc](https://wiki.opnfv.org/dpacc)
Mailing list: Opnfv-tech-discussion [dpacc]
Weekly meeting: Wednesday at 14:00pm-15:00pm UTC
[https://wiki.opnfv.org/high_availability_project_meetings](https://wiki.opnfv.org/high_availability_project_meetings)
On demand IRC discussion threads on
dpacc requirements drafting, virtio-ipsec implementation, inline acceleration etc.
denglingli@chinamobile.com
Virtio-IPsec-LA PoC Implementation

Denis Crasta, Freescale
denis.crasta@freescale.com
**Virtio-IPsec-LA PoC implementation details**

**Fastpath**
- Receives packets from virtio-net devices and does forwarding and IPsec
- Registers with Linux kernel for offload of flows, routes and SAs
- Both the above are facilitated by fastpath patch to Linux kernel

**Virtio-IPsec Frontend**
- IPsecFP uses g-API to access the virtio-IPsec device

**Virtio-IPsec Backend**
- Uses the user mode driver for the IPsec accelerator hardware

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

- FastPath
  - Receives packets from virtio-net devices and does forwarding and IPsec
  - Registers with Linux kernel for offload of flows, routes and SAs
  - Both the above are facilitated by fastpath patch to Linux kernel

- Virtio-IPsec Frontend
  - IPsecFP uses g-API to access the virtio-IPsec device

- Virtio-IPsec Backend
  - Uses the user mode driver for the IPsec accelerator hardware
Virtio-IPsec-LA PoC Setup

- Freescale LS2085RDB
  - Hosts the IPsec VNF
  - Implements Virtio-IPsec-LA acceleration

- Another laptop (LT2) is used for remote GW. IPsec implemented in Linux

- Laptop LT1 is used to generate clear traffic from LAN side.
g-API for IPSec

Data API
• `g_ipsec_la_packet_encap()`  
  Send a packet for encapsulation
• `g_ipsec_la_packet_decap()`  
  Send a packet for decapsulation
• `g_ipsec_la_multi_packet_encap()`  
  Send multiple packets for encapsulation
• `g_ipsec_la_multi_packet_decap()`  
  Send multiple packets for decapsulation

Control API
• `g_ipsec_la_capabilities_get()`  
  Get the capabilities of the underlying devices
• `g_ipsec_la_sa_add()`  
  Add SA
• `g_ipsec_la_sa_del()`  
  Delete SA
• `g_ipsec_la_sa_mod()`  
  Modify SA
• `g_ipsec_la_sa_flush()`  
  Flush SA
• `g_ipsec_la_sa_get()`  
  Read and Traversal SA
• `g_ipsec_la_notifications_hook_register()`  
  Register hooks for optional notifications such as Sequence number overflow or lifetime in kilobytes expiry etc.

Management API
• `g_ipsec_la_get_api_version()`  
  Get the API version
• `g_ipsec_la_avail_devices_getinfo()`  
  Get the information on available devices
• `g_ipsec_la_active_devices_getinfo()`  
  Get the information on active devices
• `g_ipsec_la_open()`  
  Open a device
• `g_ipsec_la_close()`  
  Close a device
• `g_ipsec_la_group_create()`  
  Create a logical group for grouping SAs
• `g_ipsec_la_group_delete()`  
  Delete a logical group