DATA PLANE DEVELOPMENT KIT

DPDK with KNI – Pushing the Performance of an SDWAN gateway to Highway Limits

Sabyasachi Sengupta, Nokia
[sabyasachi.sengupta@nuagenetworks.net]

DPDK Summit - San Jose – 2017

Linux Foundation

Agenda

- Introduction – 2 mins
  - Understanding a typical SDWAN Ecosystem, SDWAN gateway, Key Performance Requirements, Why DPDK?
- Solution Overview – 3 mins
  - Dedicated DPDK apvs OVS-DPDK, KNI
- Solution Design with DPDK – 5 mins
  - Software Architecture, High-level design, Component Design & Threading Model, Configuration Management
- The Big Picture – 3 mins
  - Addressing SDWAN gateway requirements
- Conclusion – 2 mins
  - Current status, Future Work, Credits, Further Reading & Q/A

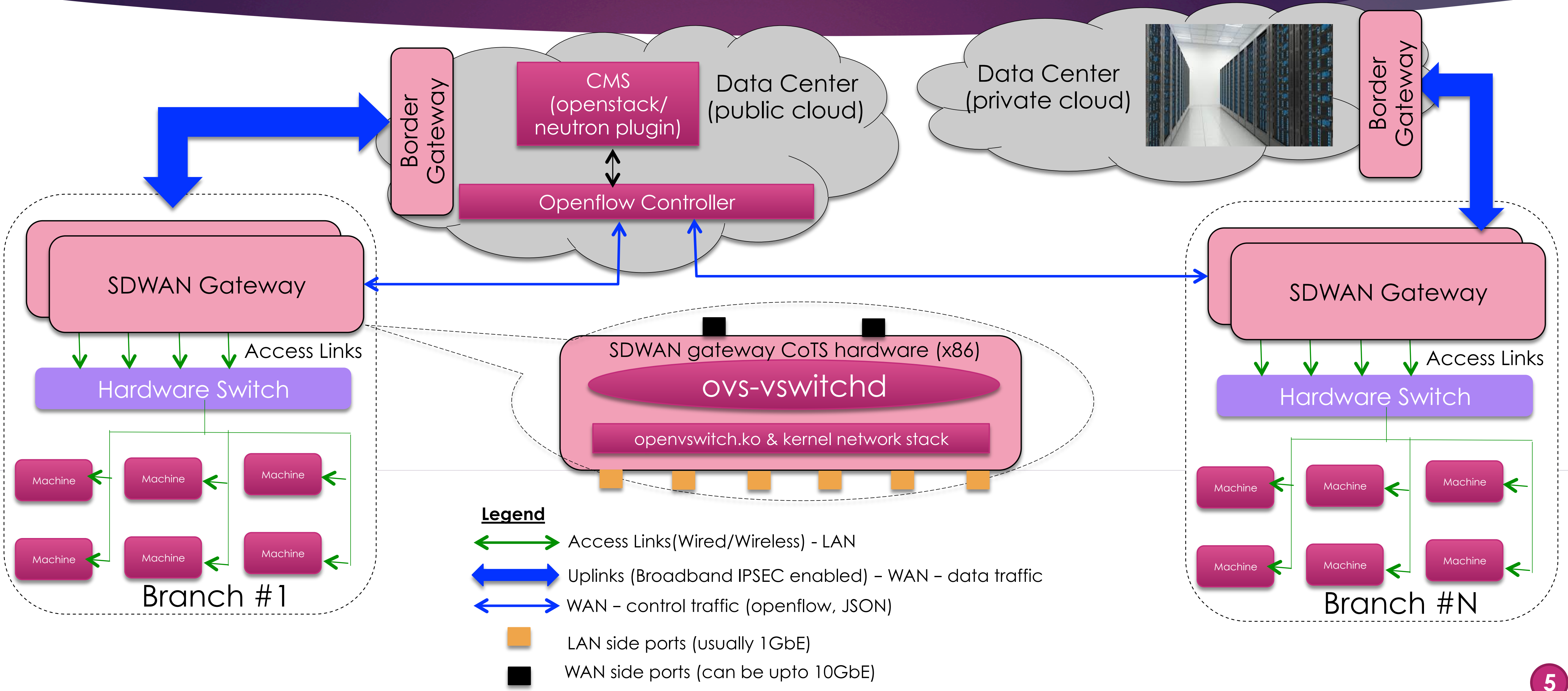
- Introduction – 2 mins
  - Understanding a typical SDWAN Ecosystem, SDWAN gateway, Key Performance Requirements, Why DPDK?
- Solution Overview
- Solution Design with DPDK
- The Big Picture
- Conclusion

Understanding the SDWAN Ecosystem

- Software Defined WAN is centered around a gateway (usually a *COTS* hardware) through which a branch connects with other branches and its Enterprise Data center through IPSEC enabled broadband.
- SDWAN gateway is centrally managed by *Zero Touch* Provisioning (ZTP) and often needs to provide high speed throughput.
- The gateway hardware comprises of:
  - One or two ports face the WAN side (aka uplink), can support high speed Internet (each port can be 10Gbps Full Duplex)
  - Remaining ports face the LAN side (aka access link), usually can be Gigabit Ethernet – connects directly to hosts or other switches/routers depending on the size of the branch
- The gateway software usually:
  - Runs Linux base OS (customized Redhat, Centos or Ubuntu)
  - Virtualization software for supporting VNFs
  - Virtual Switch (viz. Open vSwitch) – for switching packets to other overlay and underlay destinations (local and/or remote to the branch)

The SDWAN Ecosystem

A Simplified Illustration



Some SDWAN Datapath Considerations

- For an Enterprise Branch, the SDWAN gateway must be capable of high speed data transfer. For cheaper capacity, Enterprises are adding Broadband links to MPLS. Often a hybrid approach. This means:
  - Consistency of Performance – provide QoS (rate limiting and policing)
  - According to IDC, “Today 40-60% of Enterprise data is migrating between WAN to the Internet”
  - Example: Voice and Video Streaming capability across the WAN between hosts in different branches and main office
- As data is transferred across broadband, security is the key
  - Typically done through Group Key Exchange – each gateway acts as an IPSEC end point.

Why DPDK?

Limitations of Kernel based forwarding

- Using Linux Kernel for high speed data path typically has some inherent issues:
  - Linux Kernel default pagesize: 4K
    - This means for every three packets (1500 MTU) there could be a page fault. During large number of packets processed, this could introduce lot of delay.
  - No dedicated resources for packet processing
    - CPU and memory (pages) shared with rest of system
  - All ports are kernel managed
    - Packets arrive in kernel's network stack and passes through several layers of kernel before reaching virtual switch (Open vSwitch). This can introduce bottlenecks.
    - First packet given to OVS user space for openflow rules table consultation leading to more bottlenecks.
- Result: even though SDWAN gateway has 20G uplink, it cannot meet the performance requirements!

Why DPDK?

Limitations of Kernel based forwarding

- Using Linux Kernel for high speed data path typically has some inherent issues:
  - Linux Kernel default pagesize: 4K
    - This means for every three packets (1500 MTU) there could be a page fault. During large number of packets processed, this could introduce lot of delay.
  - No dedicated resources for packet processing
    - CPU and memory (pages) shared with rest of system
  - All ports are kernel managed
    - Packets arrive in kernel's network stack and passes through several layers of kernel before reaching virtual switch (Open vSwitch). This can introduce bottlenecks.
    - First packet given to OVS user space for openflow rules table consultation leading to more bottlenecks.
- Result: even though SDWAN gateway has 20G uplink, it cannot meet the performance requirements!

Advantages of DPDK

- DPDK supports larger pagesize:
  - 2M or 1G hugepages
- DPDK allows dedicated resources attached to network ports (PMD). Also memory can set aside for packet processing.
- DPDK allows packets to be received directly in the user space using PMD
- What is specifically needed for SDWAN?
  - Add IPSEC capability
  - Add QoS capability

- Introduction
- Solution Overview – 3 mins
  - Dedicated DPDK app vs OVS-DPDK, KNI
- Solution Design with DPDK
- The Big Picture
- Future Work & Conclusion

The Need for a Dedicated DPDK app

Current State of Art Analysis

DPDK

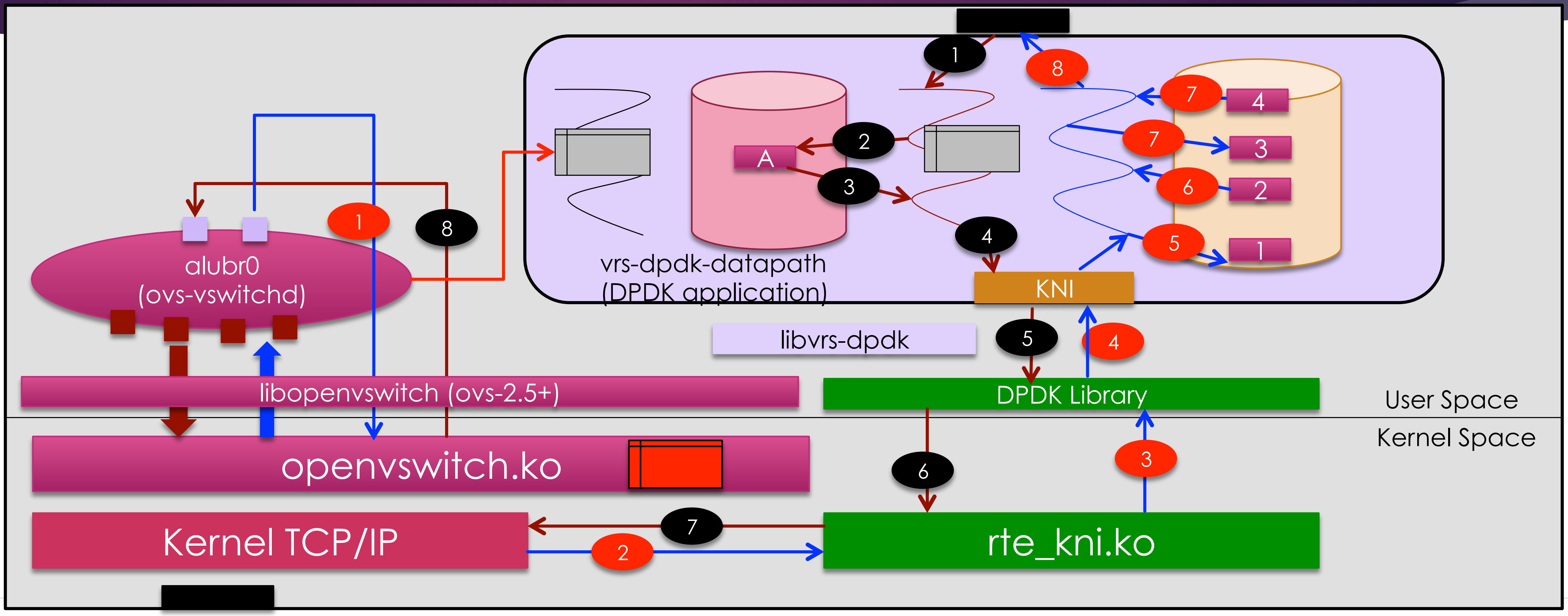
- Open vSwitch has integrated DPDK (ovs-dpdk) as an *Userspace Datapath*
  - The main bridge is configured with *datapath_type=netdev*, which indicates packets are processed in user space instead of Linux kernel
  - Devices can be added to ovsdb with *Interface type=dpdk* and subsequently a PMD thread is spawned for polling packets
  - In SDWAN environment, this means a single virtual switch application (ovs-vswitchd) will have all capabilities of slow path (first packet processing for virtual switch features) as well as fast path (subsequent packets). What if?
    - Virtual Switch app that gets periodically refreshed by SDWAN vendors for new software features have a software glitch and crashes? Can we afford to be disconnected from the gateway? What are our options?
  - What about featureset like IP-tables (firewalling), connection tracking etc? Currently these are implemented by kernel.
- Bottom-line: can we have best of both worlds?

KNI as The Missing Link!

- Kernel Network Interface is a programming technique provided by DPDK
- KNI queues allow draining of packets between DPDK app (vrs-dpdk-datapath) to/from kernel
  - A set of KNI queues (slave network devices) that are attached to a DPDK port
- Associate them with same MAC addresses
- Abstracted by dpdk_bondX device, where X = DPDK managed port ID
- Upsides:
  - Leverage OVS for all slow-path and basic virtual switching functionalities as packets arriving from WAN can be fed into kernel resident openvswitch flow tables
  - Leverage kernel for all IPtables and conntrack functionalities
- Downsides:
  - Still introduces a copy between kernel and DPDK app – cannot be avoided
  - Agreed, but experimental data shows KNIs are rather fast!!

- Introduction
- Solution Overview
- Solution Design with DPDK – 5 mins
  - Software Architecture, High-level design, Component Design & Threading Model, Configuration Management
- The Big Picture
- Conclusion

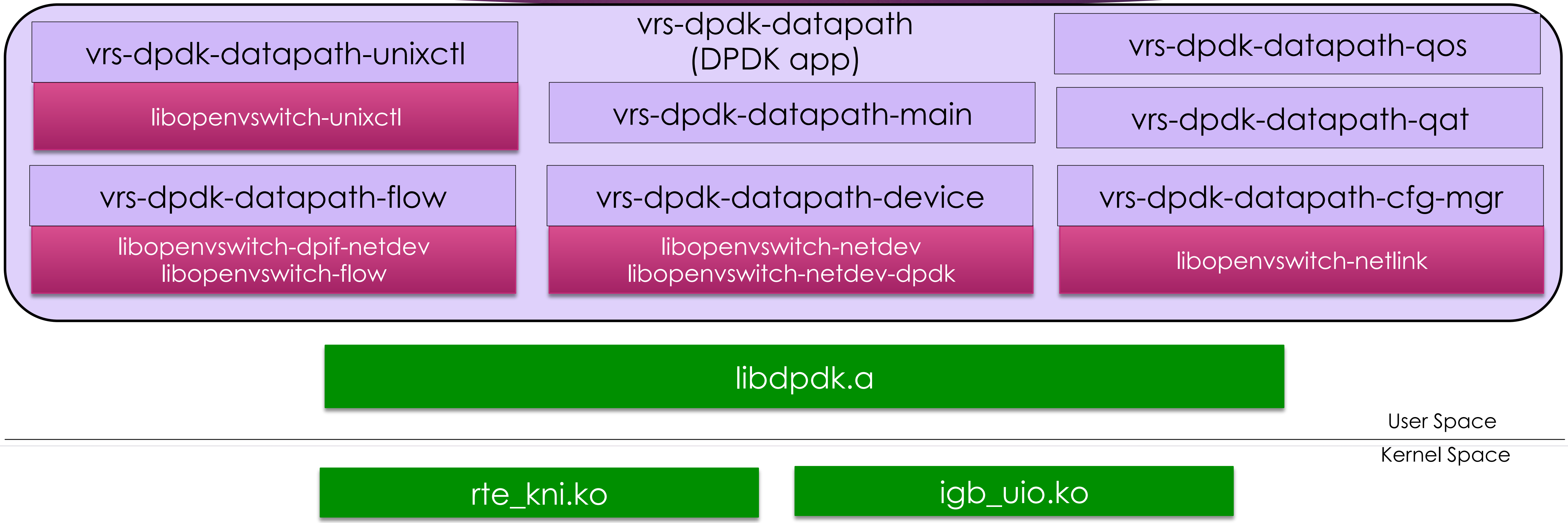
High Level Architecture



LEGEND

- Main Thread
- Receive Processing
- Transmit Processing
- NW to Access Traffic Path
- Access to NW Traffic Path
- Control Path (netlink)
- Classifier/Flow Table
- NW/A Pipeline
- A/NW Pipeline
- N NW/A Packet Hop
- N A/NW Packet Hop
- Access vLAN VPORT
- Uplink VPORT (tunnel)
- Physical Port (Access/Uplink)
- Kernel Flow Table for switching
- A QAT (ingress)
- 1 QAT (egress)
- 2 UDP/Mac Encap
- 3 QoS
- 4 Fragmentation

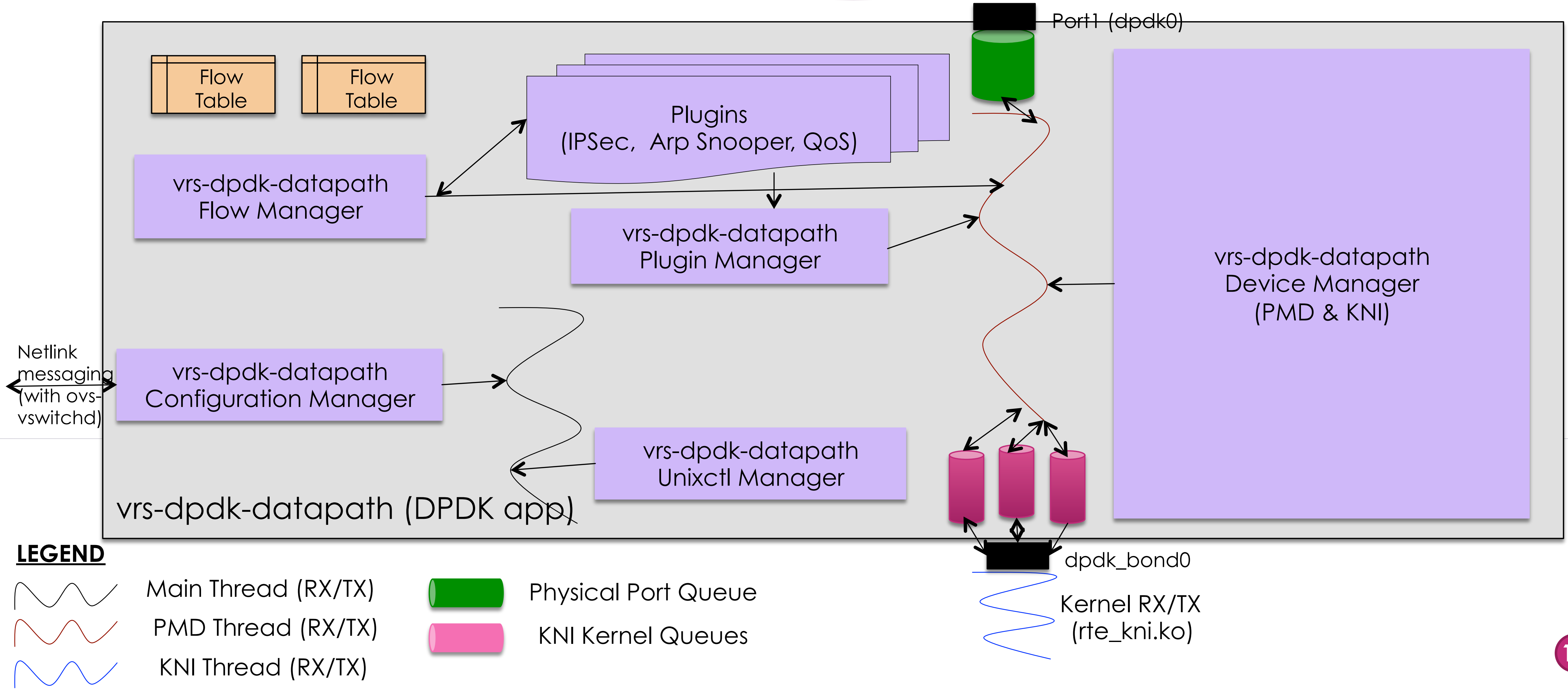
Software Architecture



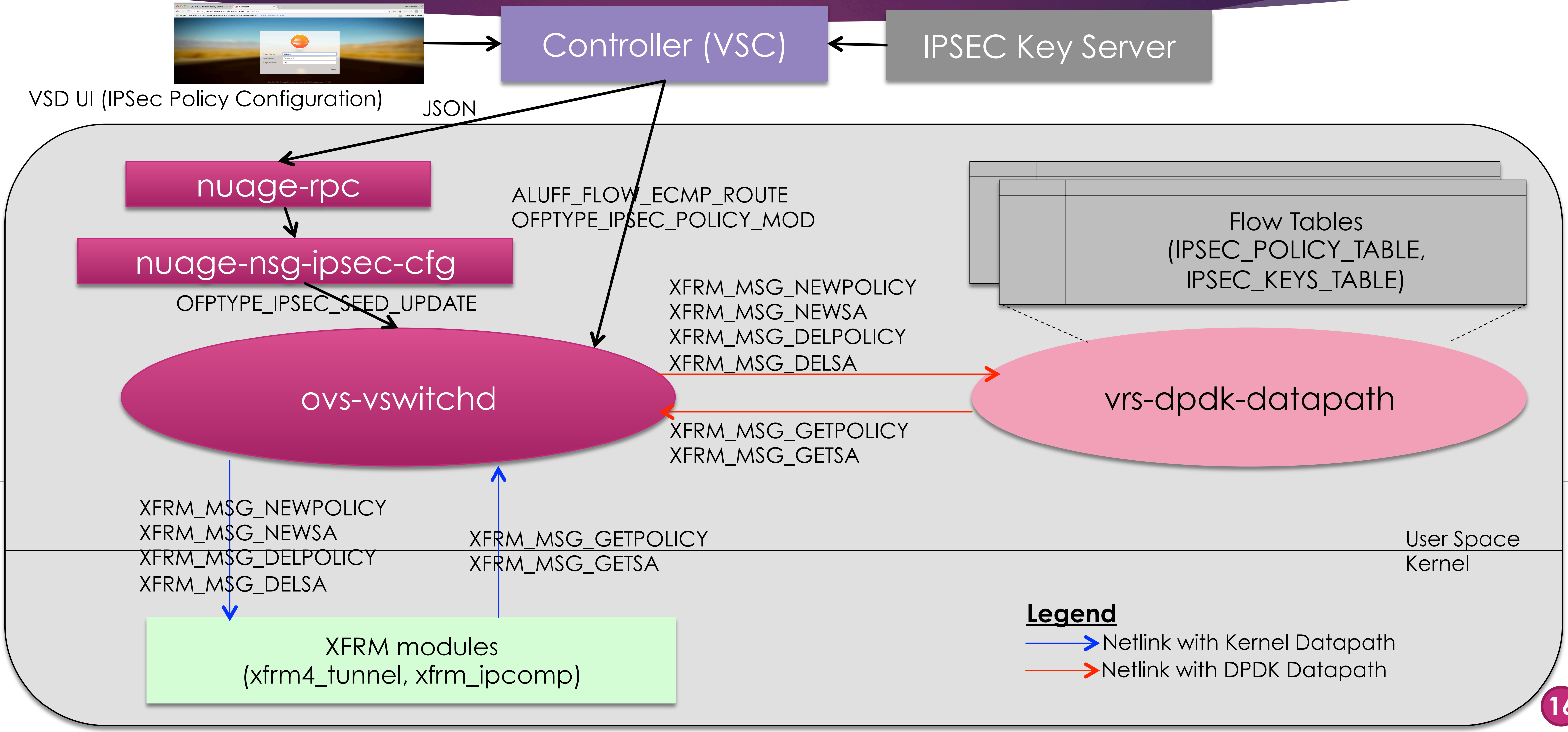
LEGEND

- vrs-dpdk-datapath component – DPDK application code (Nokia developed)
- DPDK SDK (upgraded to support DPDK-17.02 – DPDK LTS)
- Open vSwitch library code

Component Design & Threading Model



Security Configuration Management



- Introduction
- Solution Overview
- Solution Design with DPDK
- The Big Picture – 3 mins
  - Addressing SDWAN gateway requirements
- Conclusion

Meeting SDWAN gateway specific requirements

DPDK

- Enabling/Disabling DPDK on WAN ports in the gateway
  - Could be dynamically done through SDN UI by ticking off *Network Acceleration*
  - DPDK app starts, sets up hugeTLB pages, scans PCI bus and configures DPDK ports from list of whitelisted devices
  - Note: SDWAN gateways should NOT have downtimes, meaning “no reboots” or traffic loss while enabling/disabling DPDK
- SDWAN gateway underlay networking configuration
  - DPDK enabled WAN ports get their IP addresses automatically from DHCP server
  - DHCP server works seamlessly: modification in KNI infra to ensure rte_kni_alloc accepts and assigns DPDK physical port’s mac address
  - Network Manager hooks added to setup underlay IP tables and routes
- Tuning gateway for best results
  - Need to judiciously balance IRQs so as to ensure KNIs get enough cores and PMD threads get full CPU cycles
  - Other interesting configurations in KNI devices: Packet Steering Parameters (rps_cpus/xps_cpus) in /proc, txqlen, ring parameters

DPDK Profiles – Monetization opportunity!

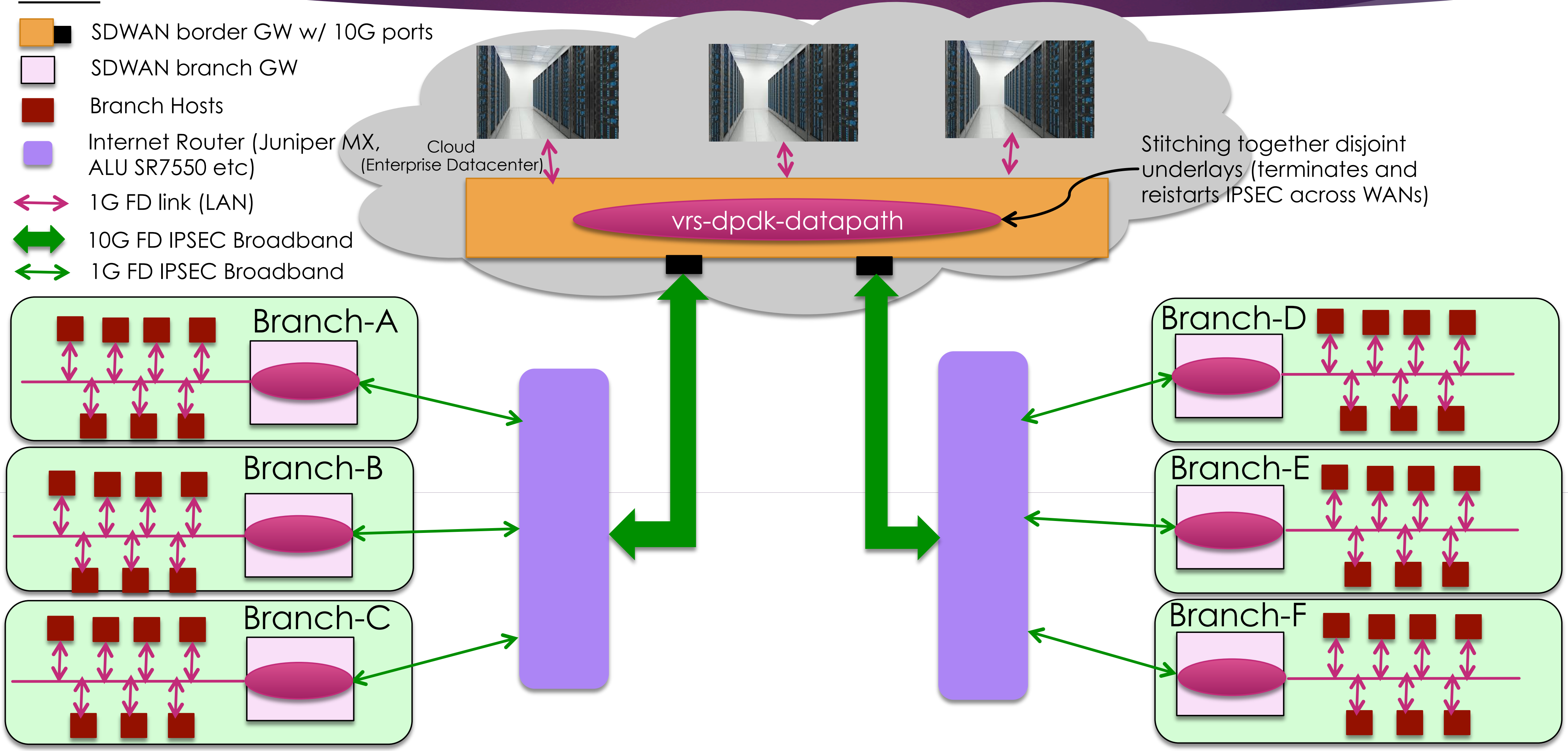
- Customers can enable *Profiles* dynamically at Cloud Director UI and add / remove CPU allocation
- Switching from one profile to another often works seamlessly without gateway reboots!
- Just a restart of the DPDK application.

Profile	# PMD / uplink	# KNI / uplink	Usage
Normal (Small)	1	1	Regular (day-to-day) processing
Accelerated (Medium)	1	3	Encrypt/decrypt happens in same PMD thread. Three KNIs drain packets from PMD. Useful for higher workload than regular profile.
Performance (Large)	2	4	One dedicated CPU (core) each for encrypt and decrypt of packets, four KNIs associated with each uplink. Excellent throughput (upto 7Gbps HD) – useful for high performance WAN traffic (voice/video).

Branch–Cloud–Branch: DPDK everywhere!!

LEGEND

- SDWAN border GW w/ 10G ports
- SDWAN branch GW
- Branch Hosts
- Internet Router (Juniper MX, ALU SR7550 etc)
- 1G FD link (LAN)
- 10G FD IPSEC Broadband
- 1G FD IPSEC Broadband



- Introduction
- Solution Overview
- Solution Design with DPDK
- The Big Picture
- Conclusion – 2 mins
  - Current status, Future Work, Credits, Further Reading & Q/A

Current State & Future Work

- Current status
  - Upto 7Gbps Half Duplex with IPSEC on 10Gbps WAN link
  - Highway: 55mph, Freeway: 65+ mph
  - That **copy** between user space and kernel space!
  - Kernel IRQ processing becomes the bottleneck after that rate
  - Still way way better than original: ~2Gbps H/D with IPSEC on 10Gbps WAN link

- Solution:
  - Move all LAN side ports to DPDK.
  - vrs-dpdk-datapath app acts as fastpath app and sends first packet to the slow path ovs-vswitchd app
  - Implement flow cache inside vrs-dpdk-datapath along with the pipeline.

Credits

- Engineering
  - Sabyasachi Sengupta – sabyasachi.sengupta@nokia.com - SDWAN ecosystem & DPDK/OVS Infrastructure
  - Paul Hong – paul.hong@nokia.com - DPDK Plugin Management Infrastructure
  - Limin Wang – limin.wang@nokia-bell-labs.com - IPSEC plugin, Fragmentation
  - Ravilochan Shamanna – ravilochan.samanna@nokia.com - IPSEC plugin
  - John Shirron – john.shirron@nokia.com - QoS plugin
  - Rohit Patil Bagli – rohit.patil-bagli@nokia.com - Functional test
  - Priyanka Kumar – priyanka.kumar@nokia.com - Functional test
  - Ankush Singh – ankush.singh@nokia.com - Performance test
- Product Management
  - Prasad Nellipudi – prasad.nellipudi@nokia.com
- Program Management
  - Raymond Zhang – raymond.zhang@nokia.com

References

- Nuage Networks SDWAN Brochure
  - http://www.nuagenetworks.net/wp-content/uploads/2015/04/PR1503009766_NN_VNS_Extensible_Wide_Area-Networking_Brochure.pdf