Implementing SDN and NFV with Intel® Architecture

Part II of series discusses how network operators, telecommunications equipment manufacturers (TEMs) and ISVs can accelerate the development of a software-focused network using open source APIs and reference designs from Intel.

**Introduction**

Part one of this four paper series detailed how software-defined networking (SDN) and a complementary initiative, network functions virtualization (NFV), are enabling a more flexible networking architecture compared to traditional approaches that use fixed-function network elements. This paper, Part II, overviews key hardware and software components used to implement a software-focused network in a virtualized environment. After reading the paper, one will be able to describe the high-level hardware and software requirements for the four architectural layers; the communication protocols linking the layers; the co-existence of physical and virtual switching; and the available Intel® reference designs and tools that can help reduce development time.

**Networking Framework**

SDN and NFV architecture combined consists of four layers, called orchestration, network applications, network controller (or controller) and node, as shown in Figure 1. The following sections cover each layer in more detail, describing their functions, the available hardware and software solutions, and the APIs that can be used for interlayer communications.

![Figure 1. An example of SDN transforming the network](image)

Industry momentum around SDN and NFV is creating a solid foundation for flexible networking.
**Orchestration layer**

The orchestration layer provides an abstraction layer between the physical network and operator network applications, such as policy and location services. The result is an application's transport requests are communicated as functional requirements instead of hardware-specific commands. This abstraction simplifies the development and deployment of network applications since they no longer bear the responsibility for configuring, provisioning or optimizing physical networking resources. Previously, these responsibilities may have required network administrators to hand-code tens of thousands of lines of configuration scattered among thousands of devices.

**Operation example**

An orchestrator has a view of all the different platforms in the network and can monitor the resource utilization (e.g., memory and CPU) and virtual machines (VMs) of each node. If a node becomes oversubscribed, the orchestrator can either allocate more computing resources to it or create another software image instance in a new VM.

**Software**

Network operators are looking to data center and enterprise technologies to help manage NFV/SDN deployments. Consequently, there already exist some open source and proprietary options, like OpenStack*, the free, open source software used by cloud-based service providers today, which is also suitable for SDN and NFV implementations. The software is available through the OpenStack Foundation, which promotes the development, distribution and adoption of the software. As the independent home for OpenStack, the Foundation has already attracted more than 7,000 individual members from 100 countries and 850 different organizations.

OpenStack is a collection of open source technologies delivering a highly-scalable, cloud operating system. It controls large pools of compute, storage and networking resources, all managed through a dashboard that gives network administrators control to provision resources through a web interface. The OpenStack cloud operating system enables enterprises and service providers to offer on-demand computing resources by provisioning and managing large networks of virtual machines. Compute resources can be accessed via APIs by developers building cloud applications, and controlled via web interfaces by administrators and users. The compute architecture is designed to scale horizontally on standard hardware, enabling the cloud economics companies have come to expect.

**Orchestration managed networking**

OpenStack communicates with the controller layer via the Quantum API, as shown in Figure 2. Quantum, which is maintained by the OpenStack Quantum project, provides the ability to build rich networking topologies and configure advanced network policies in the cloud, such as flow monitoring controls. The API enables network operators to develop innovative plug-ins, both open and closed source, to support new network capabilities.

**Hardware**

The orchestration layer runs on industry-standard, high-volume Intel® architecture-based servers.

---

**Figure 2. Software-defined networking proposed by Intel**

**Network applications layer**

The network applications layer contains network applications such as traffic engineering, load balancing and path optimization.

**Operation example**

Some network applications, like traffic shaping and policy management, manage traffic flow during peak periods, leading to improved quality of service (QoS) for most subscribers and ensuring a higher quality of experience (QoE) for those on congested networks who are paying a premium. If operators have real-time knowledge about the traffic characteristics and demand in their network, segmented by application, user and time-of-day, they can deliver customized services that provide high value to specific customers. Subscribers selecting premium rate plans rather than "best-effort" options can reliably be given an experience commensurate with the higher costs.
Software
Network applications designed to interoperate with lower layers in the network may also include support for some network controller south bound protocols such as OpenFlow.

Interlayer protocol
Network applications interface to the network controller via the north bound API.

Hardware
Like the orchestration layer, the networks applications layer runs on standard, high-volume servers.

Controller layer
The controller layer performs the control plane functions of the network and executes orchestration requests related to network resources, among other things. Network intelligence is centralized in software-based SDN controllers, which maintain a global view of the network. As a result, the network appears to applications and policy engines as a single, logical entity.

Operation example
The orchestrator requests the controller layer to create a virtual network for a given tenant. The controller provides connectivity on that virtual network to its member VMs under the direction of the orchestrator.

SDN controllers may ultimately provide complete visibility and control over the network, thus they can ensure access control, traffic engineering, quality of service, security and other policies are enforced consistently across the wired and wireless network infrastructures, including branch offices, campuses and data centers.

Software
Multiple vendors are supplying SDN controller software, all of which support OpenFlow, one of the first standard communications interfaces defined between the control and forwarding layers (i.e., node layer) of an SDN architecture.

Interlayer protocol
The optimal choice for a southbound API and Layer 2 communications protocol will depend on the specifics of the network being controlled. One possibility is OpenFlow, which allows direct access to, and manipulation of, the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based). OpenFlow incorporates switch design mechanisms with an internal flow-table and a standardized interface to add and remove flow entries. The Open Networking Foundation (ONF) is now the home of the OpenFlow specification.

Hardware
Similar to the orchestration and network applications layers, the controller layer (i.e., SDN controllers) runs on standard, high-volume servers.

Node layer
The node layer performs the data plane (i.e., forwarding) functions of the network using computing devices that have been abstracted for applications and network services, allowing them to treat the network as a logical or virtual entity. This layer supports both physical and virtual switches, thereby addressing existing and emerging switch types.

Software
Physical switches, such as top-of-rack (ToR) switches, will typically run software developed by the original equipment manufacturer (OEM) on a proprietary hardware platform. In the case of virtual switching, the software runs in either a VM or a hypervisor on a standard server. One software option is to deploy Open vSwitch (OVS), a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license and now part of the Linux kernel.

Interlayer protocol
As mentioned previously, the SDN controller uses OpenFlow and other protocols to access and manipulate the forwarding plane of node plane devices, such as switches and routers, either physical or virtual. In addition to control using OpenFlow, either vendor extensions or other APIs may be used to provide additional control over virtual or physical switches. One example would be to configure the Open vSwitch via the OVS-database interface/protocol.

Hardware
Physical switches are typically built with application-specific silicon components, like ASICs, dedicated to packet forwarding. Virtual switches run on standard, high-volume Intel architecture-based servers.

The Co-existence of Physical and Virtual Switches
Both physical and virtual switches have a place in SDN architecture. Figure 3 illustrates a hybrid SDN network that combines ToR switches (with and without OpenFlow) and virtual switches based on Open vSwitch. The ability to mix and match switch types provides investment protection for network operators.

Figure 3. Hybrid SDN Network
Why would network operators deploy both physical and virtual switches? Physical switches generally deliver faster packet forwarding throughput and lower latency; but on the other hand, they do not scale as well or as easily. For instance, virtual routers can support a larger number of tunnels and flows, and provide more deployment options, as detailed in Table 1.

**Stateful Usage Model**

Since virtual switches can also execute control applications, another advantage is their ability to support stateful traffic analysis. For instance, a virtual node can perform denial of service (DNS) detection requiring analysis on multiple packets of suspected flows. In contrast, a physical switch would depend on the SDN controller for this function if operating in OpenFlow mode.

The ability of virtual switches to handle both Layer 2 and 3 workloads opens the door to a wide range of functionality, including deep packet inspection (DPI), encryption, traffic steering and intrusion prevention, and provides additional flexibility about how the services are managed and updated.

### Table 1. Comparison of physical and virtual router

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Advantage</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fastest packet forwarding throughput and lowest latency</td>
<td>Physical switches</td>
<td>They are designed with specialty components to maximize packet forwarding performance. Physical switches provide the connectivity required at an aggregation point. A ToR switch provides physical connectivity for all of the servers in the rack.</td>
</tr>
<tr>
<td>Tunneling protocols such as VxLAN and generic routing encapsulation (GRE)</td>
<td>Virtual switches</td>
<td>Software switches have greater scaling potential, with respect to the number of supported tunnels, than physical switches. Increasing the number of tunnels requires additional CPU cycles.</td>
</tr>
<tr>
<td>Maximum number of flows</td>
<td>Virtual switches</td>
<td>They can support in excess of one million flows (typically limited by memory provisioning on the platform) compared to about 100K maximum for some physical switches.</td>
</tr>
<tr>
<td>Maximum deployment flexibility</td>
<td>Virtual switches</td>
<td>They can be deployed on any general-purpose server in the network, and capacity can be easily increased by adding more VMs.</td>
</tr>
<tr>
<td>Stateful traffic analysis</td>
<td>Virtual switches</td>
<td>They provide greater flexibility to perform control applications at the node level, which is needed for various security functions (e.g., IPsec tunneling, encryption) and flow analysis.</td>
</tr>
</tbody>
</table>

### Intel® Reference Designs for Physical and Virtual Switch Designs

Equipment manufacturers can take advantage of two Intel reference designs to develop physical and virtual switches for deployment in the SDN node layer, called respectively:

- Intel® Open Network Platform Switch Reference Design (Intel® ONP Switch Reference Design)
- Intel® Open Network Platform Server Reference Design (Intel® ONP Server Reference Design)

Both of these designs support Open Networking Software (ONS), which includes OpenFlow and a rich API designed to provide a variety of Layer 1, Layer 2 and Layer 3 features. ONS has already been integrated with select Intel switching silicon chipsets.

*Figure 4. The SDN-enabled Intel® Open Network Platform Switch Reference Design*
Intel® ONP Switch Reference Design

This reference design incorporates the necessary hardware and software components needed to build a fully-functioning ToR switch, as depicted in Figure 5. The hardware platform consists of the Intel® Ethernet Switch FM6764 for packet forwarding and an AMC module based on the Intel® Xeon® processor E5-26xx series and Intel® Communications Chipset 89xx Series for control plane functions. The software components include a software framework from Wind River* to assist customers in developing SDN solutions.

Figure 5. ToR switch based on the Intel® ONP Switch Reference Design

Switch silicon

The SDN-optimized Intel Ethernet Switch FM6764 has exceptional frame parsing capabilities and implements programmable pattern matching tables at ultra-low latencies. The switch silicon supports OpenFlow v1.0, along with extensions such as VxLAN and NVGRE.

AMC control plane module

The module is based on the Intel® Platform for Communications Infrastructure with built-in acceleration for common workloads, including packet forwarding, bulk cryptography and compression. These capabilities, available on commercial off-the-shelf (COTS) modules and servers, are a more flexible alternative to purpose-built hardware. Performance throughput of 160 million packets per second (MPPS) of L3 forwarding and 80 gigabits (Gbps) per second of IPSec acceleration have been demonstrated on servers with dual Intel Xeon processor E5-26xx series and Intel Communications Chipset 89xx Series. While the Intel® ONP Switch Reference Design is one instance of how the switch silicon and an Intel processor can be designed into a platform, other possibilities exist since the software is easily ported to other platforms with the Intel Ethernet Switch FM6764.

Software framework

The Open Network Software (ONS) is an advanced modular software platform designed to provide a variety of Layer 1, 2 and 3 features integrated with a variety of switching silicon chipsets and is developed in partnership with Wind River.

The software framework is based on:

- The Linux operating system
- Standard network protocols
- Open management APIs

The ONS is a complete network switch software environment that provides the tools and resources required to create, modify and update network layer functionality. The ONS is open and extensible, and embraces current and emerging networking technologies (such as SDN and OpenFlow) and enables third parties to augment and extend with value-added capabilities.

The modular architecture of the ONS and the open APIs enable integration with different hardware and software components (open source, third parties, or home-grown embedded control plane applications or network orchestration software) through a central controller. The ONS scales to highly distributed systems, multiple core processors and platforms, as well as small embedded single-board systems.

The ONS exposes its external interfaces using embedded database tables, which the external programmer can access using the Management API. Setup, configuration and functionality can be influenced and modified through the Management API (Figure 5) via command line interface (CLI), Simple Network Management Protocol (SNMP) v2 or v3, or XML-RPC.

APIs

The reference design implements three APIs in addition to ONS with OpenFlow.

- The management API abstracts control plane functions, and supports XML-RPC, CLI and SNMP interfaces, a switch system object model and the Linux programming model.
- The core switch API abstracts data plane functions, and supports a switch “port abstraction” model and a functional interface for data plane programming elements.
- The switch adapter API abstracts the switch hardware, and supports heterogeneous switching, and software and hardware data planes.
Intel® ONP Server Reference Design

This reference design, diagramed in Figure 6, runs on nearly any Intel® Xeon® or Intel® Core™ processor-based hardware platform. The KVM hypervisor® and Intel® Virtualization Technology (Intel® VT)® provide a flexible, high performance and robust virtualization environment. In the future, additional hypervisor options will be available. A reference implementation of a high performance version of Open vSwitch, accelerated by the Intel® Data Plane Development Kit (Intel® DPDK), is included. In addition, optimizations will be provided to facilitate remote management and integration into the orchestration infrastructure. For some workloads, the use of PCI-SIG Single Root I/O Virtualization (SR-IOV) could be used to provide direct access to virtual appliances.

Two Key Reference Design Capabilities

Fundamental to the performance and flexibility of the Intel reference designs are two key capabilities, accelerated packet forwarding and a common API between physical and virtual switches, which are described in more detail in the following sections.

Accelerated packet forwarding

The consolidation of data and control planes on a general-purpose processor has been significantly advanced by the Intel DPDK, which greatly boosts packet processing performance and throughput. Pre-integrated in Wind River Linux, the Intel DPDK provides Intel architecture-optimized libraries to accelerate L3 forwarding, yielding performance that scales linearly with the number of cores, in contrast to native Linux. The solution is supported by the Wind River development environment, further simplifying use and code debug.

The Intel DPDK contains a growing number of libraries (Figure 7), whose source code is available for developers to use and/or modify in a production network element. Likewise, there are various use case examples, like L3 forwarding, load balancing and timers, that help reduce development time. The libraries can be used to build applications based on “run-to-completion” and/or “pipeline” models, enabling the equipment provider’s application to maintain complete control. The following provides a brief description of key software components:
• The Environment Abstraction Layer (EAL) provides access to low-level resources (hardware, memory space, logical cores, etc.) through a generic interface that hides the environment specifics from the applications and libraries.

• The Memory Pool Manager allocates NUMA-aware pools of objects in memory. The pools are created in huge-page memory space to increase performance by reducing translation looksaside buffer (TLB) and cache misses.

• The Buffer Manager significantly reduces the amount of time the system spends allocating and de-allocating buffers.

• The Queue Manager implements safe lockless queues, instead of using spinlocks, that allow different software components to process packets while avoiding unnecessary wait times.

• The Ring Manager is a lockless implementation for single or multi-producer/consumer enqueue/de-queue operations, supporting bulk operations to reduce overhead for efficient passing of events, data and packet buffers.

• Flow Classification provides an efficient mechanism for generating a hash (based on tuple information) used to combine packets into flows, which enables faster processing and greater throughput.

• Poll Mode Drivers greatly speed up the packet pipeline by receiving and transmitting packets from 1 GbE and 10 GbE Ethernet controllers without the use of asynchronous, interrupt-based signaling mechanisms, which have a lot of overhead.

For more information on the Intel DPDK, please visit www.intel.com/go/dpdk.

A common API between physical and virtual switches
The Intel reference designs implement Open Networking Software (ONS), which facilitates interoperability in hybrid SDN networks and greatly simplifies SDN controller software development. ONS includes support for OpenFlow and Open vSwitch but also goes further by supporting many more features that can make networks faster and more efficient.

At the core of ONS is a well-defined object model with well-understood behaviors. The object model is defined in an XML descriptive language that describes the internal database schema and specific access methods (some of which may not be programmable) derived from the database schema.

Understanding the database schema is extremely important because it gives developers control over ONS and represents every piece of data needed by the switch northbound API. This includes ports, their structure and characteristics, and protocol interfaces like VLAN, LAG, STP, etc. All modifiable parameters and information required by the northbound API and equipment developers are represented and modeled in the database schema. There are other APIs within the virtual switch domain that will be defined as well, and as they become available in future silicon, it will be easier to incorporate offload capabilities.

ONGOING DEVELOPMENT ACTIVITIES
Moving forward, the capabilities of SDN and NFV networks could be significantly enhanced by modifications to OpenStack that provide the orchestration layer more information about the underlying functionality and capacity of devices in the node layer. For example, a recent OpenStack modification enables administrators to designate a group of compute hosts as “trusted”. These platforms establish a root of trust through measurements when the hardware and pre-launch software components are in a known good state, using hardware-based security features such as Intel® Trusted Execution Technology (Intel® TXT). Combined with an external, standalone, web-based remote attestation server, cloud providers can ensure the compute node is running software with verified measurements; thus, they can establish the foundation for a secure cloud stack. Through the trusted compute pools, cloud subscribers can request services to be run on verified compute nodes.11

Background12
OpenStack is a collection of open source projects, some of which are shown as blocks in Figure 8, that deliver a cloud operating system. There are core components, which can be used to create Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) offerings:

Object Store (Swift) stores or retrieves files (e.g., object/blob files).
Image (Glance) stores, discovers and retrieves VM images.
Compute (Nova) creates virtual servers upon demand.
Dashboard (Horizon) supports a modular web-based user interface for OpenStack services.
Identity (Keystone) authenticates and authorizes all the OpenStack services.
Network (Quantum) establishes connectivity between interface devices.
Block Storage (Cinder) provides persistent block storage to guest VMs.
Trusted compute pool example

Earlier, it was suggested that if OpenStack provided more information about the special hardware capabilities of individual nodes, network operators could more easily take advantage of these capabilities. For instance, users of cloud services requiring maximum protection for sensitive data may request their applications run in a trusted compute pool; hence, the aforementioned modification to OpenStack may be employed:

Step 1: Glance, which manages and administers VM images, is requested to load an application in a trusted compute pool. Glance broadcasts, “This VM image must be run in a trusted compute pool.”

Step 2: Nova, which creates VMs, identifies the platforms in the node layer with trusted execution technology and broadcasts, “Here is a node with that capability.”

Step 3: Another Nova module then performs the necessary matchmaking to ensure the VM image is loaded onto a platform in the trusted compute pool.

For more details about the orchestration layer and newly approved enhancements to OpenStack, see Part III of this series titled, “SDN Orchestration Layer Implementation Considerations.”

ACCELERATING SDN NETWORK DEVELOPMENT

This paper proposes a set of hardware and software components that could be used to develop an entire SDN and NFV-based network. In addition, Intel is making two reference designs available to reduce the development time and effort required to design physical and virtual switches. This solution was made possible by two fundamental innovations, the Intel DPDK and the ONS API, which accelerate packet forwarding, and create a common API between physical and virtual switches, respectively. With the objective of advancing SDN further, Intel is working with other companies and standards bodies to drive greater intelligence, capability and standardization into SDN/NFV APIs, thus enabling operators to more fully utilize their network resources.

For more information about Intel® solutions for networking and communications, visit www.intel.com/go/commsinfrastructure.

---

4Source: www.openflow.org
7Performance estimates are based on internal Intel analysis and are provided for informational purposes only.
8Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel® products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit http://www.intel.com/performance/resources/limits.htm.
9Source: www.linux-kvm.org for more information.
10Intel® Virtualization Technology (Intel® VT) requires a computer system with an enabled Intel® processor, BIOS, virtual machine monitor (VMM), and for some uses, certain platform software enabled for it. Functionality, performance, or other benefits will vary depending on hardware and software configurations and may require a BIOS update. Software applications may not be compatible with all operating systems. Please check with your application vendor.